Information Theory and Probabilistic Programming

Samuel Cheng

School of ECE
University of Oklahoma

October 10, 2019

An optimization example

- Simple economy: m prosumers, n different goods[1]
- Each individual: production $\mathbf{p}_i \in \mathbb{R}_n$, consumption $\mathbf{c}_i \in \mathbb{R}_n$
- Expense of producing "$\mathbf{p}$" for agent $i = e_i(\mathbf{p})$
- Utility (happiness) of consuming "$\mathbf{c}$" units for agent $i = u_i(\mathbf{c})$
- Maximize happiness

$$\max_{\mathbf{p}_i,\mathbf{c}_i} \sum_{i=1}^{m} (u_i(\mathbf{c}_i) - e_i(\mathbf{p}_i)) \qquad s.t. \qquad \sum_{i=1}^{m} \mathbf{c}_i = \sum_{i=1}^{m} \mathbf{p}_i$$

[1] Example borrowed from the first lecture of Prof Gordon's CMU CS 10-725

Walrasian equilibrium

$$\max_{\mathbf{p}_i,\mathbf{c}_i} \sum_{i=1}^{m} (u_i(\mathbf{c}_i) - e_i(\mathbf{p}_i)) \qquad s.t. \qquad \sum_{i=1}^{m} \mathbf{c}_i = \sum_{i=1}^{m} \mathbf{p}_i$$

- Idea: introduce price λ_j to each good j. Let the market decide
  - Price $\lambda_j \uparrow$: consumption of good $j \downarrow$, production of good $j \uparrow$
  - Price $\lambda_j \downarrow$: consumption of good $j \uparrow$, production of good $j \downarrow$
  - Can adjust price until consumption = production for each good

Algorithm: tâtonnement

Assume that the appropriate prices are found, we can ignore the equality constraint, then the problem becomes

$$\max_{\mathbf{p}_i,\mathbf{c}_i} \sum_{i=1}^{m} (u_i(\mathbf{c}_i) - e_i(\mathbf{p}_i)) \quad \Rightarrow \quad \sum_{i=1}^{m} \max_{\mathbf{p}_i,\mathbf{c}_i} (u_i(\mathbf{c}_i) - e_i(\mathbf{p}_i))$$

So we can simply optimize production and consumption of each individual independently

Algorithm 1 tâtonnement

1: **procedure** FindBestPrices
2: $\lambda \leftarrow [0, 0, \cdots, 0]$
3: **for** $k = 1, 2, \cdots$ **do**
4: Each individual solves for its c_i and p_i for the given λ
5: $\lambda \leftarrow \lambda + \delta_k \sum_i (c_i - p_i)$

Lagrange multiplier

Problem

$$\max_{\mathbf{x}} f(\mathbf{x})$$

$$g(\mathbf{x}) = 0$$

Consider $L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x})$ and let $\tilde{f}(\mathbf{x}) = \min_\lambda L(\mathbf{x}, \lambda)$.

Lagrange multiplier

Problem

$$\max_{\mathbf{x}} f(\mathbf{x})$$

$$g(\mathbf{x}) = 0$$

Consider $L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x})$ and let $\tilde{f}(\mathbf{x}) = \min_\lambda L(\mathbf{x}, \lambda)$. Note that

$$\tilde{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) \text{ if } g(\mathbf{x}) = 0 \\ -\infty \text{ otherwise} \end{cases}$$

Lagrange multiplier

Problem

$$\max_{\mathbf{x}} f(\mathbf{x})$$

$$g(\mathbf{x}) = 0$$

Consider $L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x})$ and let $\tilde{f}(\mathbf{x}) = \min_{\lambda} L(\mathbf{x}, \lambda)$. Note that

$$\tilde{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) \text{ if } g(\mathbf{x}) = 0 \\ -\infty \text{ otherwise} \end{cases}$$

Therefore, the problem is identical to $\max_{\mathbf{x}} \tilde{f}(\mathbf{x})$ or

$$\max_{\mathbf{x}} \min_{\lambda} (f(\mathbf{x}) - \lambda g(\mathbf{x})),$$

where λ is known to be the Lagrange multiplier.

Lagrange multiplier (con't)

Assume the optimum is a saddle point,

$$\max_{\mathbf{x}} \min_{\lambda} (f(\mathbf{x}) - \lambda g(\mathbf{x})) = \min_{\lambda} \max_{\mathbf{x}} (f(\mathbf{x}) - \lambda g(\mathbf{x})),$$

the R.H.S. implies

$$\nabla f(\mathbf{x}) = \lambda \nabla g(\mathbf{x})$$

Inequality constraint

Problem

$$\max_{\mathbf{x}} f(\mathbf{x})$$

$$g(\mathbf{x}) \leq 0$$

Consider $\tilde{f}(\mathbf{x}) = \min_{\lambda \geq 0}(f(\mathbf{x}) - \lambda g(\mathbf{x}))$,

Inequality constraint

Problem

$$\max_{\mathbf{x}} f(\mathbf{x})$$
$$g(\mathbf{x}) \leq 0$$

Consider $\tilde{f}(\mathbf{x}) = \min_{\lambda \geq 0}(f(\mathbf{x}) - \lambda g(\mathbf{x}))$, note that

$$\tilde{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } g(\mathbf{x}) \leq 0 \\ -\infty & \text{otherwise} \end{cases}$$

Inequality constraint

Problem

$$\max_{\mathbf{x}} f(\mathbf{x})$$
$$g(\mathbf{x}) \leq 0$$

Consider $\tilde{f}(\mathbf{x}) = \min_{\lambda \geq 0}(f(\mathbf{x}) - \lambda g(\mathbf{x}))$, note that

$$\tilde{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } g(\mathbf{x}) \leq 0 \\ -\infty & \text{otherwise} \end{cases}$$

Therefore, we can rewrite the problem as

$$\max_{\mathbf{x}} \min_{\lambda \geq 0}(f(\mathbf{x}) - \lambda g(\mathbf{x}))$$

Inequality constraint (con't)

Assume

$$\max_{\mathbf{x}} \min_{\lambda \geq 0} (f(\mathbf{x}) - \lambda g(\mathbf{x})) = \min_{\lambda \geq 0} \max_{\mathbf{x}} (f(\mathbf{x}) - \lambda g(\mathbf{x}))$$

The R.H.S. implies

$$\nabla f(\mathbf{x}) = \lambda \nabla g(\mathbf{x})$$

Inequality constraint (con't)

Assume

$$\max_{\mathbf{x}} \min_{\lambda \geq 0} (f(\mathbf{x}) - \lambda g(\mathbf{x})) = \min_{\lambda \geq 0} \max_{\mathbf{x}} (f(\mathbf{x}) - \lambda g(\mathbf{x}))$$

The R.H.S. implies

$$\nabla f(\mathbf{x}) = \lambda \nabla g(\mathbf{x})$$

Moreover, at the optimum point $(\mathbf{x}^*, \lambda^*)$, we should have the so-called "complementary slackness" condition

$$\lambda^* g(\mathbf{x}^*) = 0$$

since

$$\max_{\substack{\mathbf{x} \\ g(\mathbf{x}) \leq 0}} f(\mathbf{x}) \equiv \max_{\mathbf{x}} \min_{\lambda \geq 0} (f(\mathbf{x}) - \lambda g(\mathbf{x}))$$

Karush-Kuhn-Tucker conditions

Problem

$$\max_{\mathbf{x}} f(\mathbf{x})$$
$$g(\mathbf{x}) \le 0, \quad h(\mathbf{x}) = 0$$

Conditions

$$\begin{aligned}
\nabla f(\mathbf{x}^*) - \mu^* \nabla g(\mathbf{x}^*) - \lambda^* \nabla h(\mathbf{x}^*) &= 0 \\
g(\mathbf{x}^*) &\le 0 \\
h(\mathbf{x}^*) &= 0 \\
\mu^* &\ge 0 \\
\mu^* g(\mathbf{x}^*) &= 0
\end{aligned}$$

Overview of source coding

- The objective of "source coding" is to compress some source

Overview of source coding

- The objective of "source coding" is to compress some source
- We can think of compression as "coding". Meaning that we replace each input by a corresponding coded sequence. So encoding is just a mapping/function process

Overview of source coding

- The objective of "source coding" is to compress some source
- We can think of compression as "coding". Meaning that we replace each input by a corresponding coded sequence. So encoding is just a mapping/function process
- Without loss of generality, we can use binary domain for our coded sequence. So for each input message, it is converted to a sequence of 1s and 0s

Overview of source coding

- The objective of "source coding" is to compress some source
- We can think of compression as "coding". Meaning that we replace each input by a corresponding coded sequence. So encoding is just a mapping/function process
- Without loss of generality, we can use binary domain for our coded sequence. So for each input message, it is converted to a sequence of 1s and 0s
- Consider encoding (compressing) a sequence $x_1, x_2, \cdots$ one symbol at a time, resulting $c(x_1), c(x_2), \cdots$

Overview of source coding

- The objective of "source coding" is to compress some source
- We can think of compression as "coding". Meaning that we replace each input by a corresponding coded sequence. So encoding is just a mapping/function process
- Without loss of generality, we can use binary domain for our coded sequence. So for each input message, it is converted to a sequence of 1s and 0s
- Consider encoding (compressing) a sequence $x_1, x_2, \cdots$ one symbol at a time, resulting $c(x_1), c(x_2), \cdots$
- Denote the lengths of $x_1, x_2, \cdots$ as $l(x_1), l(x_2), \cdots$, one of the major goal is to have $E[l(X)]$ to be as small as possible

Overview of source coding

- The objective of "source coding" is to compress some source
- We can think of compression as "coding". Meaning that we replace each input by a corresponding coded sequence. So encoding is just a mapping/function process
- Without loss of generality, we can use binary domain for our coded sequence. So for each input message, it is converted to a sequence of 1s and 0s
- Consider encoding (compressing) a sequence $x_1, x_2, \cdots$ one symbol at a time, resulting $c(x_1), c(x_2), \cdots$
- Denote the lengths of $x_1, x_2, \cdots$ as $l(x_1), l(x_2), \cdots$, one of the major goal is to have $E[l(X)]$ to be as small as possible
- However, we want to make sure that we can losslessly decode the message also!

Uniquely decodable code

- To ensure that we can recover message without loss, we must make sure that no message share the same codeword

Uniquely decodable code

- To ensure that we can recover message without loss, we must make sure that no message share the same codeword
- We say a code is "singular" (broken) if $c(x_1) = c(x_2)$ for some different x_1 and x_2

Uniquely decodable code

- To ensure that we can recover message without loss, we must make sure that no message share the same codeword
- We say a code is "singular" (broken) if $c(x_1) = c(x_2)$ for some different x_1 and x_2
- Even when a code is not "singular", we still cannot guarantee that we can always recover the original message losslessly, consider 4 different possible input symbols a, b, c, d and an encoding map $c(\cdot)$:
    - $a \mapsto 0, b \mapsto 1, c \mapsto 10, d \mapsto 11$
    - What should be the message for 1110?

Uniquely decodable code

- To ensure that we can recover message without loss, we must make sure that no message share the same codeword
- We say a code is "singular" (broken) if $c(x_1) = c(x_2)$ for some different x_1 and x_2
- Even when a code is not "singular", we still cannot guarantee that we can always recover the original message losslessly, consider 4 different possible input symbols a, b, c, d and an encoding map $c(\cdot)$:
    - $a \mapsto 0, b \mapsto 1, c \mapsto 10, d \mapsto 11$
    - What should be the message for 1110?
        - *dba*? Or *bbba*?

Uniquely decodable code

- To ensure that we can recover message without loss, we must make sure that no message share the same codeword
- We say a code is "singular" (broken) if $c(x_1) = c(x_2)$ for some different x_1 and x_2
- Even when a code is not "singular", we still cannot guarantee that we can always recover the original message losslessly, consider 4 different possible input symbols a, b, c, d and an encoding map $c(\cdot)$:
  - $a \mapsto 0, b \mapsto 1, c \mapsto 10, d \mapsto 11$
  - What should be the message for 1110?
    - *dba*? Or *bbba*?
- So it is not sufficient to just have $c(\cdot)$ to map to different output for each input. Let's overload the notation $c(\cdot)$ a little bit and for any message sequence $\mathbf{x} = x_1, x_2, \cdots, x_n$, encode sequence $x_1, x_2, \cdots, x_n$ to $c(\mathbf{x}) = c(x_1, x_2, \cdots, x_n) = c(x_1)c(x_2)\cdots c(x_n)$

Uniquely decodable code

- To ensure that we can recover message without loss, we must make sure that no message share the same codeword
- We say a code is "singular" (broken) if $c(x_1) = c(x_2)$ for some different x_1 and x_2
- Even when a code is not "singular", we still cannot guarantee that we can always recover the original message losslessly, consider 4 different possible input symbols a, b, c, d and an encoding map $c(\cdot)$:
  - $a \mapsto 0, b \mapsto 1, c \mapsto 10, d \mapsto 11$
  - What should be the message for 1110?
    - *dba*? Or *bbba*?
- So it is not sufficient to just have $c(\cdot)$ to map to different output for each input. Let's overload the notation $c(\cdot)$ a little bit and for any message sequence $\mathbf{x} = x_1, x_2, \cdots, x_n$, encode sequence $x_1, x_2, \cdots, x_n$ to $c(\mathbf{x}) = c(x_1, x_2, \cdots, x_n) = c(x_1)c(x_2)\cdots c(x_n)$
  - We say $c(\mathbf{x})$ is uniquely decodable if all input sequences map to different outputs

Prefix-free code

- For practical purpose, we would like to be able to decode a symbol "once it is available". Consider a code with map
  - $a \mapsto 10, b \mapsto 00, c \mapsto 11, d \mapsto 110$

Prefix-free code

- For practical purpose, we would like to be able to decode a symbol "once it is available". Consider a code with map
  - $a \mapsto 10, b \mapsto 00, c \mapsto 11, d \mapsto 110$
  - One can show that it is uniquely decodable. However, consider an input sequence $cbbb \mapsto 11000000$

Prefix-free code

- For practical purpose, we would like to be able to decode a symbol "once it is available". Consider a code with map
  - $a \mapsto 10, b \mapsto 00, c \mapsto 11, d \mapsto 110$
  - One can show that it is uniquely decodable. However, consider an input sequence $cbbb \mapsto 11000000$
  - When the decoder read the first 3 bits, it is not able to determine if the first input symbol is c or d

Prefix-free code

- For practical purpose, we would like to be able to decode a symbol "once it is available". Consider a code with map
  - $a \mapsto 10, b \mapsto 00, c \mapsto 11, d \mapsto 110$
  - One can show that it is uniquely decodable. However, consider an input sequence $cbbb \mapsto 11000000$
  - When the decoder read the first 3 bits, it is not able to determine if the first input symbol is c or d
  - Actually, it will be until the decoder read the last bit that it will be able to confirm that the first input symbol is c. It is definitely not something very desirable

Prefix-free code

- For practical purpose, we would like to be able to decode a symbol "once it is available". Consider a code with map
  - $a \mapsto 10, b \mapsto 00, c \mapsto 11, d \mapsto 110$
  - One can show that it is uniquely decodable. However, consider an input sequence $cbbb \mapsto 11000000$
  - When the decoder read the first 3 bits, it is not able to determine if the first input symbol is c or d
  - Actually, it will be until the decoder read the last bit that it will be able to confirm that the first input symbol is c. It is definitely not something very desirable
- Instead, for a mapping $a \mapsto 1, b \mapsto 01, c \mapsto 001, d \mapsto 0001$, I will argue that we can always decode a symbol "once it is available"

Prefix-free code

- For practical purpose, we would like to be able to decode a symbol "once it is available". Consider a code with map
  - $a \mapsto 10, b \mapsto 00, c \mapsto 11, d \mapsto 110$
  - One can show that it is uniquely decodable. However, consider an input sequence $cbbb \mapsto 11000000$
  - When the decoder read the first 3 bits, it is not able to determine if the first input symbol is c or d
  - Actually, it will be until the decoder read the last bit that it will be able to confirm that the first input symbol is c. It is definitely not something very desirable
- Instead, for a mapping $a \mapsto 1, b \mapsto 01, c \mapsto 001, d \mapsto 0001$, I will argue that we can always decode a symbol "once it is available"
  - Note that the catch is that there is no codeword being the "prefix" of another codeword
  - We call such code a prefix-free code or an instantaneous code

Kraft's Inequality

- How do we know if a length profile for a code is possible?
- Kraft's inequality: Consider a length profile $l_1, l_2, \cdots, l_K$, there exists a uniquely decodable code for symbols $x_1, x_2, \cdots, x_K$ such that $l(x_1) = l_1$, $l(x_2) = l_2, \cdots, l(x_K) = l_K$ if and only if $\sum_{k=1}^{K} 2^{-l_k} \leq 1$

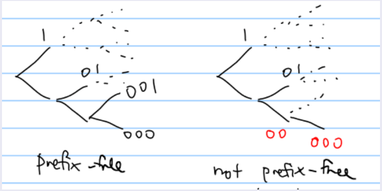
Kraft's Inequality

- How do we know if a length profile for a code is possible?
- Kraft's inequality: Consider a length profile $l_1, l_2, \cdots, l_K$, there exists a uniquely decodable code for symbols $x_1, x_2, \cdots, x_K$ such that $l(x_1) = l_1$, $l(x_2) = l_2, \cdots, l(x_K) = l_K$ if and only if $\sum_{k=1}^{K} 2^{-l_k} \leq 1$

Intuition

Consider # "descendants" of each codeword at the "l_{max}"-level, then for prefix-free code, we have

$$\sum_{k=1}^{K} 2^{l_{max}-l} \leq 2^{l_{max}}$$



Kraft's Inequality

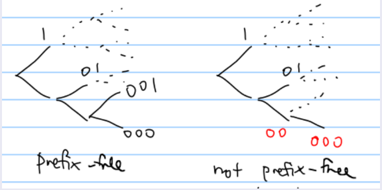
- How do we know if a length profile for a code is possible?
- Kraft's inequality: Consider a length profile $l_1, l_2, \cdots, l_K$, there exists a uniquely decodable code for symbols $x_1, x_2, \cdots, x_K$ such that $l(x_1) = l_1,\ l(x_2) = l_2, \cdots, l(x_K) = l_K$ if and only if $\sum_{k=1}^{K} 2^{-l_k} \leq 1$

Intuition

Consider # "descendants" of each codeword at the "l_{max}"-level, then for prefix-free code, we have

$$\sum_{k=1}^{K} 2^{l_{max}-l} \leq 2^{l_{max}}$$

$$\Rightarrow \sum_{k=1}^{K} 2^{-l_k} \leq 1$$



Forward Proof

Given $l_1, l_2, \cdots, l_K$ satisfy $\sum_{k=1}^{K} 2^{-l_k} \leq 1$, we can assign nodes on a tree as previous slides. More precisely,

- Assign i-th node as a node at level l_i, then cross out all its descendants
- Repeat the procedure for i from 1 to K
- We know that there are sufficient tree nodes to be assigned since the Kraft's inequaltiy is satisfied

The corresponding code is apparently prefix-free and thus is uniquely decodable

Converse Proof

Consider message from coding k symbols $\mathbf{x} = x_1, x_2, \cdots, x_k$

$$
\begin{aligned}
\left(\sum_{x\in\mathcal{X}} 2^{-l(x)}\right)^k &= \left(\sum_{x_1\in\mathcal{X}} 2^{-l(x_1)}\right)\left(\sum_{x_2\in\mathcal{X}} 2^{-l(x_2)}\right)\cdots\left(\sum_{x_k\in\mathcal{X}} 2^{-l(x_k)}\right) \\
&= \sum_{x_1,x_2,\cdots,x_k\in\mathcal{X}^k} 2^{-(l(x_1)+l(x_2)+\cdots+l(x_k))} \\
&= \sum_{\mathbf{x}\in\mathcal{X}^k} 2^{-l(\mathbf{x})}
\end{aligned}
$$

Converse Proof

Consider message from coding k symbols $\mathbf{x} = x_1, x_2, \cdots, x_k$

$$\left(\sum_{x\in\mathcal{X}} 2^{-l(x)}\right)^k = \left(\sum_{x_1\in\mathcal{X}} 2^{-l(x_1)}\right)\left(\sum_{x_2\in\mathcal{X}} 2^{-l(x_2)}\right)\cdots\left(\sum_{x_k\in\mathcal{X}} 2^{-l(x_k)}\right)$$

$$= \sum_{x_1,x_2,\cdots,x_k\in\mathcal{X}^k} 2^{-(l(x_1)+l(x_2)+\cdots+l(x_k))}$$

$$= \sum_{\mathbf{x}\in\mathcal{X}^k} 2^{-l(\mathbf{x})} = \sum_{m=1}^{kl_{max}} a(m)2^{-m},$$

where $a(m)$ is the number of codeword with length m. However, for the code to be uniquely decodable, $a(m) \leq 2^m$, where 2^m is the number of available codewords with length m.

Converse Proof

Consider message from coding k symbols $\mathbf{x} = x_1, x_2, \cdots, x_k$

$$\left(\sum_{x\in\mathcal{X}} 2^{-l(x)}\right)^k = \left(\sum_{x_1\in\mathcal{X}} 2^{-l(x_1)}\right)\left(\sum_{x_2\in\mathcal{X}} 2^{-l(x_2)}\right)\cdots\left(\sum_{x_k\in\mathcal{X}} 2^{-l(x_k)}\right)$$
$$= \sum_{x_1,x_2,\cdots,x_k\in\mathcal{X}^k} 2^{-(l(x_1)+l(x_2)+\cdots+l(x_k))}$$
$$= \sum_{\mathbf{x}\in\mathcal{X}^k} 2^{-l(\mathbf{x})} = \sum_{m=1}^{kl_{max}} a(m)2^{-m},$$

where $a(m)$ is the number of codeword with length m. However, for the code to be uniquely decodable, $a(m) \leq 2^m$, where 2^m is the number of available codewords with length m. Therefore,

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

Converse Proof

Consider message from coding k symbols $\mathbf{x} = x_1, x_2, \cdots, x_k$

$$\left(\sum_{x\in\mathcal{X}} 2^{-l(x)}\right)^k = \left(\sum_{x_1\in\mathcal{X}} 2^{-l(x_1)}\right)\left(\sum_{x_2\in\mathcal{X}} 2^{-l(x_2)}\right)\cdots\left(\sum_{x_k\in\mathcal{X}} 2^{-l(x_k)}\right)$$
$$= \sum_{x_1,x_2,\cdots,x_k\in\mathcal{X}^k} 2^{-(l(x_1)+l(x_2)+\cdots+l(x_k))}$$
$$= \sum_{\mathbf{x}\in\mathcal{X}^k} 2^{-l(\mathbf{x})} = \sum_{m=1}^{kl_{max}} a(m)2^{-m},$$

where $a(m)$ is the number of codeword with length m. However, for the code to be uniquely decodable, $a(m) \leq 2^m$, where 2^m is the number of available codewords with length m. Therefore,

$$\sum_{x\in\mathcal{X}} 2^{-l(x)} \leq (kl_{max})^{1/k} \approx 1 \text{ as } k \to \infty$$

Minimum rate required to compress a source

$$min_{l_1,l_2,\cdots,l_K} \sum_{k=1}^{K} p_k l_k \text{ subject to } \sum_{k=1}^{K} 2^{-l_k} \le 1 \text{ and } l_1,\cdots,l_K \ge 0$$

$$\equiv max_{l_1,l_2,\cdots,l_K} - \sum_{k=1}^{K} p_k l_k \text{ subject to } \sum_{k=1}^{K} 2^{-l_k} - 1 \le 0 \text{ and } -l_1,\cdots,-l_K \le 0$$

KKT conditions

$$-\nabla\left(\sum_{k=1}^{K} p_k l_k\right) - \mu_0 \nabla\left(\sum_{k=1}^{K} 2^{-l_k} - 1\right) + \sum_{k=1}^{K} \mu_k \nabla l_k = 0$$

Minimum rate required to compress a source

$$min_{l_1,l_2,\cdots,l_K} \sum_{k=1}^{K} p_k l_k \text{ subject to } \sum_{k=1}^{K} 2^{-l_k} \leq 1 \text{ and } l_1,\cdots,l_K \geq 0$$

$$\equiv max_{l_1,l_2,\cdots,l_K} -\sum_{k=1}^{K} p_k l_k \text{ subject to } \sum_{k=1}^{K} 2^{-l_k} - 1 \leq 0 \text{ and } -l_1,\cdots,-l_K \leq 0$$

KKT conditions

$$-\nabla\left(\sum_{k=1}^{K} p_k l_k\right) - \mu_0 \nabla\left(\sum_{k=1}^{K} 2^{-l_k} - 1\right) + \sum_{k=1}^{K} \mu_k \nabla l_k = 0$$

$$\sum_{k=1}^{K} 2^{-l_k} - 1 \leq 0, \quad l_1,\cdots,l_K \geq 0, \quad \mu_0,\mu_1,\cdots,\mu_K \geq 0$$

Minimum rate required to compress a source

$$min_{l_1,l_2,\cdots,l_K} \sum_{k=1}^{K} p_k l_k \text{ subject to } \sum_{k=1}^{K} 2^{-l_k} \leq 1 \text{ and } l_1,\cdots,l_K \geq 0$$

$$\equiv max_{l_1,l_2,\cdots,l_K} -\sum_{k=1}^{K} p_k l_k \text{ subject to } \sum_{k=1}^{K} 2^{-l_k} - 1 \leq 0 \text{ and } -l_1,\cdots,-l_K \leq 0$$

KKT conditions

$$-\nabla\left(\sum_{k=1}^{K} p_k l_k\right) - \mu_0 \nabla\left(\sum_{k=1}^{K} 2^{-l_k} - 1\right) + \sum_{k=1}^{K} \mu_k \nabla l_k = 0$$

$$\sum_{k=1}^{K} 2^{-l_k} - 1 \leq 0, \quad l_1,\cdots,l_K \geq 0, \quad \mu_0,\mu_1,\cdots,\mu_K \geq 0$$

$$\mu_0\left(\sum_{k=1}^{K} 2^{-l_k} - 1\right) = 0, \quad \mu_k l_k = 0$$

Minimum rate required to compress a source

Since we expect $l_k > 0$, $\mu_k = 0$.

Minimum rate required to compress a source

Since we expect $l_k > 0$, $\mu_k = 0$. Expand the first equation, we get

$$-p_j + \mu_0 2^{-l_j} \log 2 = 0 \Rightarrow 2^{-l_j} = \frac{p_j}{\mu_0 \log 2}$$

Minimum rate required to compress a source

Since we expect $l_k > 0$, $\mu_k = 0$. Expand the first equation, we get

$$-p_j + \mu_0 2^{-l_j} \log 2 = 0 \Rightarrow 2^{-l_j} = \frac{p_j}{\mu_0 \log 2}$$

And by $\sum_{k=1}^{K} 2^{-l_k} \leq 1$, we have

$$\sum_{k=1}^{K} \frac{p_j}{\mu_0 \log 2} = \frac{1}{\mu_0 \log 2} \leq 1 \Rightarrow \mu_0 \geq \frac{1}{\log 2}$$

Minimum rate required to compress a source

Since we expect $l_k > 0$, $\mu_k = 0$. Expand the first equation, we get

$$-p_j + \mu_0 2^{-l_j} \log 2 = 0 \Rightarrow 2^{-l_j} = \frac{p_j}{\mu_0 \log 2}$$

And by $\sum_{k=1}^{K} 2^{-l_k} \leq 1$, we have

$$\sum_{k=1}^{K} \frac{p_j}{\mu_0 \log 2} = \frac{1}{\mu_0 \log 2} \leq 1 \Rightarrow \mu_0 \geq \frac{1}{\log 2}$$

Note that as $\mu_0 \downarrow$, $\frac{p_j}{\mu_0 \log 2} \uparrow$ and $l_j \downarrow$.

Minimum rate required to compress a source

Since we expect $l_k > 0$, $\mu_k = 0$. Expand the first equation, we get

$$-p_j + \mu_0 2^{-l_j} \log 2 = 0 \Rightarrow 2^{-l_j} = \frac{p_j}{\mu_0 \log 2}$$

And by $\sum_{k=1}^{K} 2^{-l_k} \leq 1$, we have

$$\sum_{k=1}^{K} \frac{p_j}{\mu_0 \log 2} = \frac{1}{\mu_0 \log 2} \leq 1 \Rightarrow \mu_0 \geq \frac{1}{\log 2}$$

Note that as $\mu_0 \downarrow$, $\frac{p_j}{\mu_0 \log 2} \uparrow$ and $l_j \downarrow$. Therefore, if we want to decrease code rate, we should reduce μ_0 as much as possible. Thus, take $\mu_0 = \frac{1}{\log 2}$. Then $2^{-l_j} = p_j \Rightarrow l_j = -\log_2 p_j$. Thus, the minimum rate becomes

$$\sum_{k=1}^{K} p_k l_k = -\sum_{k=1}^{K} p_k \log_2 p_k \triangleq H(p_1, \cdots, p_K)$$

Review

- Kraft's inequality: $\sum_{k=1}^{K} 2^{-l_k} \leq 1$

Review

- Kraft's inequality: $\sum_{k=1}^{K} 2^{-l_k} \leq 1$
  - We showed that given a code "length-profile", we can always find a prefix-free code if the profile satisfies Kraft's inequality

Review

- Kraft's inequality: $\sum_{k=1}^{K} 2^{-l_k} \leq 1$
  - We showed that given a code "length-profile", we can always find a prefix-free code if the profile satisfies Kraft's inequality
  - Conversely, if Kraft's inequality is not satisfies, any code with that profile is not uniquely decodable $\Rightarrow$ trash

Review

- Kraft's inequality: $\sum_{k=1}^{K} 2^{-l_k} \leq 1$
  - We showed that given a code "length-profile", we can always find a prefix-free code if the profile satisfies Kraft's inequality
  - Conversely, if Kraft's inequality is not satisfies, any code with that profile is not uniquely decodable $\Rightarrow$ trash
- A proof of Source Coding Theorem: if we minimize the expected code length subject to the Kraft's inequality, the minimum "code rate" is equal to the entropy of the source.

Review

- Kraft's inequality: $\sum_{k=1}^{K} 2^{-l_k} \leq 1$
  - We showed that given a code "length-profile", we can always find a prefix-free code if the profile satisfies Kraft's inequality
  - Conversely, if Kraft's inequality is not satisfies, any code with that profile is not uniquely decodable $\Rightarrow$ trash
- A proof of Source Coding Theorem: if we minimize the expected code length subject to the Kraft's inequality, the minimum "code rate" is equal to the entropy of the source.
  - We cannot compress a source losslessly below its entropy

Review

- Kraft's inequality: $\sum_{k=1}^{K} 2^{-l_k} \leq 1$
  - We showed that given a code "length-profile", we can always find a prefix-free code if the profile satisfies Kraft's inequality
  - Conversely, if Kraft's inequality is not satisfies, any code with that profile is not uniquely decodable $\Rightarrow$ trash
- A proof of Source Coding Theorem: if we minimize the expected code length subject to the Kraft's inequality, the minimum "code rate" is equal to the entropy of the source.
  - We cannot compress a source losslessly below its entropy
  - On the other hand, since Kraft's inequality guarantee existence of code, we should be able to find code to achieve the theoretical limit

Review

- Kraft's inequality: $\sum_{k=1}^{K} 2^{-l_k} \leq 1$
  - We showed that given a code "length-profile", we can always find a prefix-free code if the profile satisfies Kraft's inequality
  - Conversely, if Kraft's inequality is not satisfies, any code with that profile is not uniquely decodable $\Rightarrow$ trash
- A proof of Source Coding Theorem: if we minimize the expected code length subject to the Kraft's inequality, the minimum "code rate" is equal to the entropy of the source.
  - We cannot compress a source losslessly below its entropy
  - On the other hand, since Kraft's inequality guarantee existence of code, we should be able to find code to achieve the theoretical limit
- However, the proof we discussed was not constructive. How can we actually design a code to compress arbitrarily close to the theoretical limit?

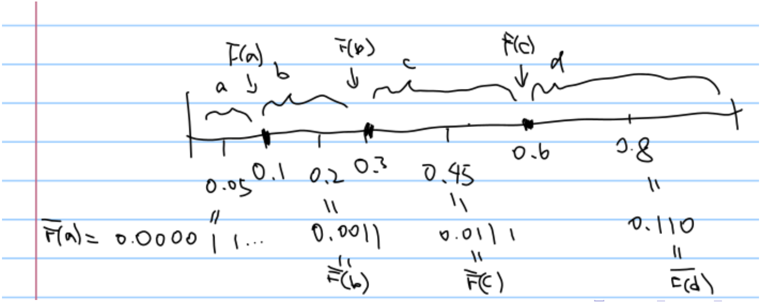
Shannon-Fano-Elias code

Key idea

Each codeword corresponds to an intervel of $[0,1]$

Example

110 corresponds to $[0.110, 0.110\dot{1}] = [0.11, 0.111) = [0.75, 0.875)$



Shannon-Fano-Elias code

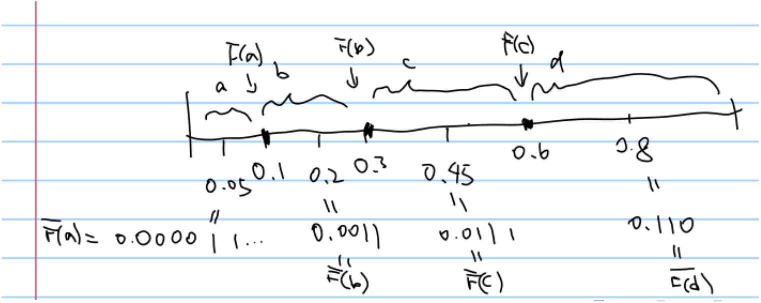
Key idea

Each codeword corresponds to an intervel of $[0,1]$

Example

110 corresponds to $[0.110, 0.110\underset{\cdot}{1}] = [0.11, 0.111) = [0.75, 0.875)$

011 corresponds to $[0.011, 0.011\underset{\cdot}{1}] = [0.011, 0.1) = [0.375, 0.5)$



Observations

Remark (Observation 1)

Let $l(x) = |c(x)|$ *be the length of the SFE codeword, and let* $u(x)$ *be the corresponding interval. Then, the length of the interval* $|u(x)| = 2^{-l(x)}$

Observations

Remark (Observation 1)

Let $l(x) = |c(x)|$ *be the length of the SFE codeword, and let* $u(x)$ *be the corresponding interval. Then, the length of the interval* $|u(x)| = 2^{-l(x)}$

Remark (Observation 2)

If $u(x_1)$ *and* $u(x_2)$ *do not overlap, then* $c(x_1)$ *and* $c(x_2)$ *cannot be prefix of one another*

Observations

Remark (Observation 1)

Let $l(x) = |c(x)|$ be the length of the SFE codeword, and let $u(x)$ be the corresponding interval. Then, the length of the interval $|u(x)| = 2^{-l(x)}$

Remark (Observation 2)

If $u(x_1)$ and $u(x_2)$ do not overlap, then $c(x_1)$ and $c(x_2)$ cannot be prefix of one another

Proof of Observation 2.

$A \Rightarrow B \equiv \neg B \Rightarrow \neg A$. We will show instead if $c(x_1)$ and $c(x_2)$ are prefix of one another, then $u(x_1)$ and $u(x_2)$ overlap. WLOG, assume $c(x_1)$ is a prefix of $c(x_2)$, the lower boundary of $u(x_1)$ is below the lower boundary of $u(x_2)$ and yet the upper boundary of $u(x_1)$ is above the upper boundary of $u(x_2)$. Thus, $u(x_2) \subseteq u(x_1)$ and hence $u(x_1)$ and $u(x_2)$ overlap each other $\square$

Example

Consider a source that
$p(x_1) = 0.25, p(x_2) = 0.25, p(x_3) = 0.2, p(x_4) = 0.15, p(x_5) = 0.15$

x	$p(x)$	$F(x)$	$\overline{F}(x)$	$\overline{F}(x)$ in Binary	$l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	Codeword
1	0.25	0.25	0.125	0.001	3	001
2	0.25	0.5	0.375	0.011	3	011
3	0.2	0.7	0.6	$0.1\overline{0011}$	4	1001
4	0.15	0.85	0.775	$0.110\overline{0011}$	4	1100
5	0.15	1.0	0.925	$0.111\overline{0110}$	4	1110

Property

- The length of the codeword of x is $\lceil \log_2 \frac{1}{p(x)} \rceil + 1$. This ensures that the "code interval" of each codeword does not overlap

Property

- The length of the codeword of x is $\lceil \log_2 \frac{1}{p(x)} \rceil + 1$. This ensures that the "code interval" of each codeword does not overlap
- Recall from observation 1, SFE code is prefix-free $\rightarrow$ uniquely decodable

Property

- The length of the codeword of x is $\lceil \log_2 \frac{1}{p(x)} \rceil + 1$. This ensures that the "code interval" of each codeword does not overlap
- Recall from observation 1, SFE code is prefix-free $\rightarrow$ uniquely decodable
  - If a codeword is prefix of another (say 10 and 1010), the corresponding intervals must overlap each other (consider $[0.10, 0.11)$ and $[0.101, 0.11)$)

Property

- The length of the codeword of x is $\lceil \log_2 \frac{1}{p(x)} \rceil + 1$. This ensures that the "code interval" of each codeword does not overlap
- Recall from observation 1, SFE code is prefix-free $\rightarrow$ uniquely decodable
  - If a codeword is prefix of another (say 10 and 1010), the corresponding intervals must overlap each other (consider $[0.10, 0.11)$ and $[0.101, 0.11)$)
  - Since no codeword can overlap in SFE, no code word can be prefix of another

Property

- The length of the codeword of x is $\lceil \log_2 \frac{1}{p(x)} \rceil + 1$. This ensures that the "code interval" of each codeword does not overlap
- Recall from observation 1, SFE code is prefix-free $\to$ uniquely decodable
  - If a codeword is prefix of another (say 10 and 1010), the corresponding intervals must overlap each other (consider $[0.10, 0.11)$ and $[0.101, 0.11)$)
  - Since no codeword can overlap in SFE, no code word can be prefix of another
- Average code rate is upper bounded by $H(X) + 2$

$$\sum_{x\in\mathcal{X}} p(x) l(x) = \sum_{x\in\mathcal{X}} p(x) \left(\left\lceil \log_2 \frac{1}{p(x)} \right\rceil + 1 \right)$$
$$\leq \sum_{x\in\mathcal{X}} p(x) \left(\log_2 \frac{1}{p(x)} + 2 \right) = H(X) + 2$$

"Symbol grouping" trick

- Let's consider two symbols as a super-symbol and compress the pair at each time with SFE code
- The code rate is bounded by $H(X_S) + 2$, where

"Symbol grouping" trick

- Let's consider two symbols as a super-symbol and compress the pair at each time with SFE code
- The code rate is bounded by $H(X_S) + 2$, where

$$H(X_S) = -\sum_{x_1, x_2 \in \mathcal{X}^2} p(x_1, x_2) \log_2 p(x_1, x_2)$$

"Symbol grouping" trick

- Let's consider two symbols as a super-symbol and compress the pair at each time with SFE code
- The code rate is bounded by $H(X_S)+2$, where

$$\begin{aligned} H(X_S) &= -\sum_{x_1,x_2\in\mathcal{X}^2} p(x_1,x_2)\log_2 p(x_1,x_2) \\ &= -\sum_{x_1,x_2\in\mathcal{X}^2} p(x_1,x_2)\log_2(p(x_1)p(x_2)) \end{aligned}$$

"Symbol grouping" trick

- Let's consider two symbols as a super-symbol and compress the pair at each time with SFE code
- The code rate is bounded by $H(X_S)+2$, where

$$
\begin{aligned}
H(X_S) &= -\sum_{x_1,x_2\in\mathcal{X}^2} p(x_1,x_2)\log_2 p(x_1,x_2) \\
&= -\sum_{x_1,x_2\in\mathcal{X}^2} p(x_1,x_2)\log_2 (p(x_1)p(x_2)) \\
&= -\sum_{x_1,x_2\in\mathcal{X}^2} p(x_1,x_2)\log_2 p(x_1) - \sum_{x_1,x_2\in\mathcal{X}^2} p(x_1,x_2)\log_2 p(x_2)
\end{aligned}
$$

"Symbol grouping" trick

- Let's consider two symbols as a super-symbol and compress the pair at each time with SFE code
- The code rate is bounded by $H(X_S)+2$, where

$$\begin{aligned}
H(X_S) &= -\sum_{x_1,x_2\in\mathcal{X}^2} p(x_1,x_2)\log_2 p(x_1,x_2)\\
&= -\sum_{x_1,x_2\in\mathcal{X}^2} p(x_1,x_2)\log_2(p(x_1)p(x_2))\\
&= -\sum_{x_1,x_2\in\mathcal{X}^2} p(x_1,x_2)\log_2 p(x_1) - \sum_{x_1,x_2\in\mathcal{X}^2} p(x_1,x_2)\log_2 p(x_2)\\
&= -\sum_{x_1\in\mathcal{X}} p(x_1)\log_2 p(x_1) - \sum_{x_2\in\mathcal{X}} p(x_2)\log_2 p(x_2)\\
&= 2H(X)
\end{aligned}$$

Therefore, the code rate per original symbol is upper bounded by

$$\frac{1}{2}(H(X_S)+2) = H(X)+1$$

Forward proof of Source Coding Theorem

In theory, we can group as many symbols as we want (we want do it in practice, why?), say we group N symbols at a time and compress it using SFE code.

Forward proof of Source Coding Theorem

In theory, we can group as many symbols as we want (we want do it in practice, why?), say we group N symbols at a time and compress it using SFE code. The code rate per original symbol is upper bounded by

$$\frac{1}{N}(H(X_S)+2)=\frac{1}{N}(NH(X)+2)=H(X)+\frac{2}{N}$$

Forward proof of Source Coding Theorem

In theory, we can group as many symbols as we want (we want do it in practice, why?), say we group N symbols at a time and compress it using SFE code. The code rate per original symbol is upper bounded by

$$\frac{1}{N}(H(X_S)+2)=\frac{1}{N}(NH(X)+2)=H(X)+\frac{2}{N}$$

Therefore as long as a given rate $R>H(X)$, we can always find a large enough N such that the code rate using the "grouping trick" and SFE code is below R. This concludes the forward proof

Entropy for discrete random variable

Von Neumman to Shannon

"You should call it entropy for two reasons: first because that is what the formula is in statistical mechanics but second and more important, as nobody knows what entropy is, whenever you use the term you will always be at an advantage!" -John von Neumman

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) = E[-\log p(X)]$$

Entropy for discrete random variable

Von Neumman to Shannon

"You should call it entropy for two reasons: first because that is what the formula is in statistical mechanics but second and more important, as nobody knows what entropy is, whenever you use the term you will always be at an advantage!" -John von Neumman

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) = E[-\log p(X)]$$

- From the expression, it suggests that there is $\log \frac{1}{p(x)}$ bits of information for the outcome x

Entropy for discrete random variable

Von Neumman to Shannon

"You should call it entropy for two reasons: first because that is what the formula is in statistical mechanics but second and more important, as nobody knows what entropy is, whenever you use the term you will always be at an advantage!" -John von Neumman

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) = E[-\log p(X)]$$

- From the expression, it suggests that there is $\log \frac{1}{p(x)}$ bits of information for the outcome x
- This actually comes with no surprise! Consider a uniform random variable with 4 outcomes, each outcome will have probalility $1/4 = 0.25$ of happening it. And to represent each outcome, we need $\log 4 = \log \frac{1}{0.25}$ bits

Entropy for discrete random variable

Von Neumman to Shannon

"You should call it entropy for two reasons: first because that is what the formula is in statistical mechanics but second and more important, as nobody knows what entropy is, whenever you use the term you will always be at an advantage!" -John von Neumman

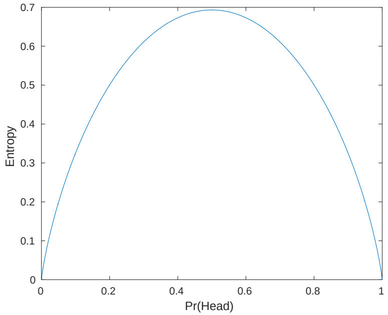
$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) = E[-\log p(X)]$$

- From the expression, it suggests that there is $\log\frac{1}{p(x)}$ bits of information for the outcome x
- This actually comes with no surprise! Consider a uniform random variable with 4 outcomes, each outcome will have probalility $1/4 = 0.25$ of happening it. And to represent each outcome, we need $\log 4 = \log\frac{1}{0.25}$ bits
- A less likely event has "more" information and requires more bits to store. $H(X)$ is just the average number of bits required

Biased coin with $Pr(Head) = p$

$$H(X) = -Pr(Head)\log Pr(Head) - Pr(Tail)\log Pr(Tail)$$
$$= -p\log p - (1-p)\log(1-p)$$

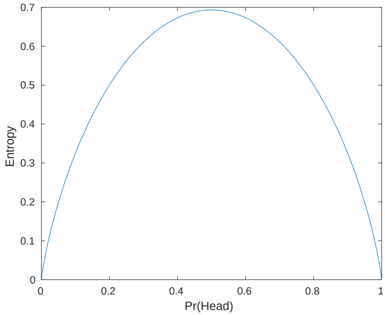
- Entropy is largest (=1) when $p = 0.5$
- Entropy is 0 when $p = 0$ or $p = 1$



Biased coin with $Pr(Head) = p$

$$H(X) = -Pr(Head)\log Pr(Head) - Pr(Tail)\log Pr(Tail)$$
$$= -p\log p - (1-p)\log(1-p)$$

- Entropy is largest (=1) when $p = 0.5$
- Entropy is 0 when $p = 0$ or $p = 1$
- Entropy can be interpreted as *the average uncertainty of the outcome* or *the amount of information "gained" after the outcome is revealed*



Differential entropy

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) = E[-\log p(X)]$$

The definition makes little sense for a continuous X. Since the probability of an outcome x is always 0, we may define instead the differential entropy for X as

$$h(X) = -\int_{x\in\mathcal{X}} p(x)\log p(x)dx$$

where $p(x)$ is now the pdf rather than the pmf

Differential entropy

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) = E[-\log p(X)]$$

The definition makes little sense for a continuous X. Since the probability of an outcome x is always 0, we may define instead the differential entropy for X as

$$h(X) = -\int_{x\in\mathcal{X}} p(x)\log p(x)dx = E[-\log p(x)],$$

where $p(x)$ is now the pdf rather than the pmf

Differential entropy of common distributions

Uniform Distribution

If $p(X) = \begin{cases} 1/a & 0 \leq x \leq a \\ 0 & \text{otherwise} \end{cases}$

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

Differential entropy of common distributions

Uniform Distribution

If $p(X) = \begin{cases} 1/a & 0 \le x \le a \\ 0 & \text{otherwise} \end{cases}$

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

Exponential distribution

For exponentially distributed $T \sim Exp(\lambda)$,

$$h(T) = E[-\log p(T)] = E\left[-\log\left(\lambda \exp(-\lambda T)\right)\right]$$

Differential entropy of common distributions

Uniform Distribution

If $p(X) = \begin{cases} 1/a & 0 \leq x \leq a \\ 0 & \text{otherwise} \end{cases}$

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

Exponential distribution

For exponentially distributed $T \sim Exp(\lambda)$,

$$\begin{aligned} h(T) &= E[-\log p(T)] = E\left[-\log\left(\lambda \exp(-\lambda T)\right)\right] \\ &= E\left[\lambda T - \log \lambda\right] \end{aligned}$$

Differential entropy of common distributions

Uniform Distribution

If $p(X) = \begin{cases} 1/a & 0 \leq x \leq a \\ 0 & \text{otherwise} \end{cases}$

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

Exponential distribution

For exponentially distributed $T \sim Exp(\lambda)$,

$$\begin{aligned} h(T) &= E[-\log p(T)] = E\left[-\log\left(\lambda \exp(-\lambda T)\right)\right] \\ &= E\left[\lambda T - \log \lambda\right] \\ &= 1 - \log \lambda \end{aligned}$$

Differential entropy of common distributions

Univariate Normal distribution

For univariate normally distributed $X \sim \mathcal{N}(\mu, \sigma^2)$,

$$h(X) = E[-\log p(X)]$$

Differential entropy of common distributions

Univariate Normal distribution

For univariate normally distributed $X \sim \mathcal{N}(\mu, \sigma^2)$,

$$h(X) = E[-\log p(X)] = E\left[-\log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\frac{-(X-\mu)^2}{2\sigma^2}\right)\right]$$

Differential entropy of common distributions

Univariate Normal distribution

For univariate normally distributed $X \sim \mathcal{N}(\mu, \sigma^2)$,

$$h(X) = E[-\log p(X)] = E\left[-\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\exp\frac{-(X-\mu)^2}{2\sigma^2}\right)\right]$$

$$= E\left[\log\sqrt{2\pi\sigma^2} + \frac{(X-\mu)^2}{2\sigma^2}\log e\right]$$

Differential entropy of common distributions

Univariate Normal distribution

For univariate normally distributed $X \sim \mathcal{N}(\mu, \sigma^2)$,

$$\begin{aligned} h(X) &= E[-\log p(X)] = E\left[-\log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\frac{-(X-\mu)^2}{2\sigma^2}\right)\right] \\ &= E\left[\log\sqrt{2\pi\sigma^2} + \frac{(X-\mu)^2}{2\sigma^2}\log e\right] \\ &= \log\sqrt{2\pi\sigma^2} + \frac{1}{2}\log e \\ &= \log\sqrt{2\pi e\sigma^2} \end{aligned}$$

N.B. $h(X)$ only depends on σ^2 and is independent of μ as one would expect

Multivariate Normal distribution

For N-dim multivariate normal distributed $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$,

$$h(\mathbf{X}) = E[-\log p(\mathbf{X})]$$
$$= -E\left[\log\left(\frac{1}{\sqrt{\det(2\pi\Sigma)}}\exp\left(-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{X}-\boldsymbol{\mu})\right)\right)\right]$$

Multivariate Normal distribution

For N-dim multivariate normal distributed $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$,

$$\begin{aligned} h(\mathbf{X}) &= E[-\log p(\mathbf{X})] \\ &= -E\left[\log\left(\frac{1}{\sqrt{\det(2\pi\Sigma)}}\exp\left(-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{X}-\boldsymbol{\mu})\right)\right)\right] \\ &= \log\sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2} E\left[\sum_{i,j}(X_i-\mu_i)\left[\Sigma^{-1}\right]_{i,j}(X_j-\mu_j)\right] \end{aligned}$$

Multivariate Normal distribution

For N-dim multivariate normal distributed $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$,

$$
\begin{aligned}
h(\mathbf{X}) &= E[-\log p(\mathbf{X})] \\
&= -E\left[\log\left(\frac{1}{\sqrt{\det(2\pi\Sigma)}}\exp\left(-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{X}-\boldsymbol{\mu})\right)\right)\right] \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2}E\left[\sum_{i,j}(X_i-\mu_i)\left[\Sigma^{-1}\right]_{i,j}(X_j-\mu_j)\right] \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2}\sum_{i,j}\left[\Sigma^{-1}\right]_{i,j}E\left[(X_j-\mu_j)(X_i-\mu_i)\right]
\end{aligned}
$$

Multivariate Normal distribution

For N-dim multivariate normal distributed $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$,

$$
\begin{aligned}
h(\mathbf{X}) &= E[-\log p(\mathbf{X})] \\
&= -E\left[\log\left(\frac{1}{\sqrt{\det(2\pi\Sigma)}}\exp\left(-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{X}-\boldsymbol{\mu})\right)\right)\right] \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2}E\left[\sum_{i,j}(X_i-\mu_i)\left[\Sigma^{-1}\right]_{i,j}(X_j-\mu_j)\right] \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2}\sum_{i,j}\left[\Sigma^{-1}\right]_{i,j}E\left[(X_j-\mu_j)(X_i-\mu_i)\right] \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2}\sum_{i,j}\left[\Sigma^{-1}\right]_{i,j}\Sigma_{j,i}
\end{aligned}
$$

Multivariate Normal distribution

For N-dim multivariate normal distributed $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$,

$$
\begin{aligned}
h(\mathbf{X}) &= E[-\log p(\mathbf{X})] \\
&= -E\left[\log\left(\frac{1}{\sqrt{\det(2\pi\Sigma)}}\exp\left(-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{X}-\boldsymbol{\mu})\right)\right)\right] \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2}E\left[\sum_{i,j}(X_i-\mu_i)\left[\Sigma^{-1}\right]_{i,j}(X_j-\mu_j)\right] \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2}\sum_{i,j}\left[\Sigma^{-1}\right]_{i,j}E\left[(X_j-\mu_j)(X_i-\mu_i)\right] \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2}\sum_{i,j}\left[\Sigma^{-1}\right]_{i,j}\Sigma_{j,i} \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{N\log e}{2} = \log\sqrt{e^N\det(2\pi\Sigma)}
\end{aligned}
$$

Multivariate Normal distribution

For N-dim multivariate normal distributed $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$,

$$\begin{aligned}
h(\mathbf{X}) &= E[-\log p(\mathbf{X})] \\
&= -E\left[\log\left(\frac{1}{\sqrt{\det(2\pi\Sigma)}}\exp\left(-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{X}-\boldsymbol{\mu})\right)\right)\right] \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2}E\left[\sum_{i,j}(X_i-\mu_i)\left[\Sigma^{-1}\right]_{i,j}(X_j-\mu_j)\right] \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2}\sum_{i,j}\left[\Sigma^{-1}\right]_{i,j}E\left[(X_j-\mu_j)(X_i-\mu_i)\right] \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{\log e}{2}\sum_{i,j}\left[\Sigma^{-1}\right]_{i,j}\Sigma_{j,i} \\
&= \log\sqrt{\det(2\pi\Sigma)} + \frac{N\log e}{2} = \log\sqrt{e^N\det(2\pi\Sigma)} = \log\sqrt{\det(2\pi e\Sigma)}
\end{aligned}$$

Differential entropy and entropy

How differential entropy is related to its discrete counterpart?

- Consider a continuous random variable X
- Let X^{Δ} is a "quantized" version of it with quantization stepsize of Δ

$$H(X^{\Delta}) = \sum -p_{X^{\Delta}}(x^{\Delta}) \log p_{X^{\Delta}}(x^{\Delta})$$

Differential entropy and entropy

How differential entropy is related to its discrete counterpart?

- Consider a continuous random variable X
- Let X^Δ is a "quantized" version of it with quantization stepsize of Δ

$$H(X^\Delta) = \sum -p_{X^\Delta}(x^\Delta) \log p_{X^\Delta}(x^\Delta) \approx \sum -p_X(x^\Delta)\Delta \log(p_X(x^\Delta)\Delta)$$

Differential entropy and entropy

How differential entropy is related to its discrete counterpart?

- Consider a continuous random variable X
- Let X^Δ is a "quantized" version of it with quantization stepsize of Δ

$$H(X^\Delta) = \sum -p_{X^\Delta}(x^\Delta)\log p_{X^\Delta}(x^\Delta) \approx \sum -p_X(x^\Delta)\Delta\log(p_X(x^\Delta)\Delta)$$
$$\approx \int -p_X(x)\log(p_X(x)\Delta)dx$$

Differential entropy and entropy

How differential entropy is related to its discrete counterpart?

- Consider a continuous random variable X
- Let X^Δ is a "quantized" version of it with quantization stepsize of Δ

$$
\begin{aligned}
H(X^\Delta) &= \sum -p_{X^\Delta}(x^\Delta)\log p_{X^\Delta}(x^\Delta) \approx \sum -p_X(x^\Delta)\Delta\log(p_X(x^\Delta)\Delta) \\
&\approx \int -p_X(x)\log(p_X(x)\Delta)dx \\
&= \int -p_X(x)\log p_X(x) - \int p_X(x)\log\Delta dx \\
&= h(X) - \log\Delta
\end{aligned}
$$

Example

Consider the processing time of a packet follow an exponential distribution with an average of 1 ms. If we want to store the time with the precision of 0.01 ms, about how many bits are needed to store the result?

Example

Consider the processing time of a packet follow an exponential distribution with an average of 1 ms. If we want to store the time with the precision of 0.01 ms, about how many bits are needed to store the result?

Answer

- The processing time T follows an exponential distribution with parameter $\lambda = 1/1 = 1ms^{-1}$

Example

Consider the processing time of a packet follow an exponential distribution with an average of 1 ms. If we want to store the time with the precision of 0.01 ms, about how many bits are needed to store the result?

Answer

- The processing time T follows an exponential distribution with parameter $\lambda = 1/1 = 1ms^{-1}$
- The corresponding differential entropy $h(T) = 1 - \log(\lambda) = 1$

Example

Consider the processing time of a packet follow an exponential distribution with an average of 1 ms. If we want to store the time with the precision of 0.01 ms, about how many bits are needed to store the result?

Answer

- The processing time T follows an exponential distribution with parameter $\lambda = 1/1 = 1ms^{-1}$
- The corresponding differential entropy $h(T) = 1 - \log(\lambda) = 1$
- If we want to store with precision of 0.01 ms, we need $h(T) - \log 0.01 \approx 7.64bits$

Lower bound of entropy

$H(X) \geq 0$

Since $p(X) \leq 1$, $-\log p(X) \geq 0$, therefore

$$H(X) = E[-\log p(X)] \geq 0$$

After all, $H(X)$ represents the required bits to compress the source X

Lower bound of entropy

$H(X) \geq 0$

Since $p(X) \leq 1$, $-\log p(X) \geq 0$, therefore

$$H(X) = E[-\log p(X)] \geq 0$$

After all, $H(X)$ represents the required bits to compress the source X

Caveat

It does NOT need to be true for differential entropy. It is possible that

$$h(X) < 0$$

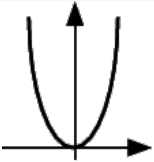
For example, for a uniformly distributed X from 0 to 0.5,

$$h(X) = \log 0.5 = -1$$

Jensen's Inequality

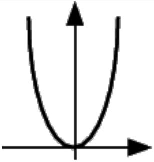
For a convex (bowl-shape) function f

$$E[f(X)] \geq f(E[X])$$



convex
function

Jensen's Inequality



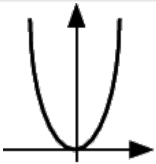
For a convex (bowl-shape) function f

$$E[f(X)] \geq f(E[X])$$

Let us consider X with only two outcomes x_1 and x_2 with probabilities p and $1-p$. Easy to see that

$$E[f(X)] = pf(x_1) + (1-p)f(x_2) \geq f(px_1 + (1-p)x_2) = f(E[X])$$

Jensen's Inequality



For a convex (bowl-shape) function f

$$E[f(X)] \geq f(E[X])$$

Let us consider X with only two outcomes x_1 and x_2 with probabilities p and $1-p$. Easy to see that

$$E[f(X)] = pf(x_1) + (1-p)f(x_2) \geq f(px_1 + (1-p)x_2) = f(E[X])$$

Result can be extended to variables with more than two outcomes easily

Upper bound of entropy

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

$$H(X) = E[-\log p(X)] = E\left[\log \frac{1}{p(X)}\right]$$

Upper bound of entropy

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

$$H(X) = E[-\log p(X)] = E\left[\log \frac{1}{p(X)}\right]$$
$$\leq \log E\left[\frac{1}{p(X)}\right] \qquad \text{(by Jensen's inequality)}$$

Upper bound of entropy

$H(X) \le \log |\mathcal{X}|$

$$
\begin{aligned}
H(X) &= E[-\log p(X)] = E\left[\log \frac{1}{p(X)}\right] \\
&\le \log E\left[\frac{1}{p(X)}\right] \qquad \text{(by Jensen's inequality)} \\
&= \log \sum_{x \in \mathcal{X}} p(x) \frac{1}{p(x)} = \log |\mathcal{X}|
\end{aligned}
$$

N.B. The upper bound is attained when the distribution is uniform

Upper bound of entropy

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

$$
\begin{aligned}
H(X) = E[-\log p(X)] &= E\left[\log \frac{1}{p(X)}\right] \\
&\leq \log E\left[\frac{1}{p(X)}\right] \qquad \text{(by Jensen's inequality)} \\
&= \log \sum_{x \in \mathcal{X}} p(x) \frac{1}{p(x)} = \log |\mathcal{X}|
\end{aligned}
$$

N.B. The upper bound is attained when the distribution is uniform

Examples

You should know this bound long alone. Think of the maximum number of bits needed:

- to store the outcome of flipping a coin: $\log 2 = 1$ bit
- to store the outcome of throwing a dice: $\log 6 \leq 3$ bits

Review

- Source coding theorem: For an independent and identically distributed (i.i.d.) discrete memoryless source (DMS) X, we can always compress it with no less than $H(X)$ bits per input symbol, where $H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) = E[-\log p(X)]$

Review

- Source coding theorem: For an independent and identically distributed (i.i.d.) discrete memoryless source (DMS) X, we can always compress it with no less than $H(X)$ bits per input symbol, where $H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) = E[-\log p(X)]$
- Jensen's inequality: For a convex (bowl-shape) function f $E[f(X)] \geq f(E[X])$. Similarly $E[g(X)] \leq g(E[X])$ for a concave g

Review

- Source coding theorem: For an independent and identically distributed (i.i.d.) discrete memoryless source (DMS) X, we can always compress it with no less than $H(X)$ bits per input symbol, where $H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) = E[-\log p(X)]$
- Jensen's inequality: For a convex (bowl-shape) function f $E[f(X)] \geq f(E[X])$. Similarly $E[g(X)] \leq g(E[X])$ for a concave g
- For continuous random variable X, the differential entropy is given by $h(X) = -\int_{x\in\mathcal{X}} p(x)\log p(x)dx = E[-\log p(x)]$

Review

- Source coding theorem: For an independent and identically distributed (i.i.d.) discrete memoryless source (DMS) X, we can always compress it with no less than $H(X)$ bits per input symbol, where $H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) = E[-\log p(X)]$
- Jensen's inequality: For a convex (bowl-shape) function f $E[f(X)] \geq f(E[X])$. Similarly $E[g(X)] \leq g(E[X])$ for a concave g
- For continuous random variable X, the differential entropy is given by $h(X) = -\int_{x\in\mathcal{X}} p(x)\log p(x)dx = E[-\log p(x)]$
- For a quantized version of continuous X, $H(X_\Delta) = h(X) - \log\Delta$

Review

- Source coding theorem: For an independent and identically distributed (i.i.d.) discrete memoryless source (DMS) X, we can always compress it with no less than $H(X)$ bits per input symbol, where $H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) = E[-\log p(X)]$
- Jensen's inequality: For a convex (bowl-shape) function f $E[f(X)] \geq f(E[X])$. Similarly $E[g(X)] \leq g(E[X])$ for a concave g
- For continuous random variable X, the differential entropy is given by $h(X) = -\int_{x\in\mathcal{X}} p(x)\log p(x)dx = E[-\log p(x)]$
- For a quantized version of continuous X, $H(X_\Delta) = h(X) - \log\Delta$
- For multivariate normal $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$,

$$h(\boldsymbol{X}) = \log\sqrt{\det(2\pi e\Sigma)}$$

Upper bound of differential entropy

$$h(X) \leq \log E\left[\frac{1}{p(X)}\right] = \log \int_{x \in \mathcal{X}} p(x) \frac{1}{p(x)} dx = \log |\mathcal{X}|$$

- The expression still makes sense but it is not useful usually since the sampling space can be unbounded $|\mathcal{X}| = \infty$ (for example, normally distributed X)

Upper bound of differential entropy

$$h(X) \leq \log E\left[\frac{1}{p(X)}\right] = \log \int_{x\in\mathcal{X}} p(x)\frac{1}{p(x)}dx = \log|\mathcal{X}|$$

- The expression still makes sense but it is not useful usually since the sampling space can be unbounded $|\mathcal{X}| = \infty$ (for example, normally distributed X)
- Thus it makes much more sense to consider upper bound of a differential entropy constrained on the variance of the variable (why not constrained on mean?)

Upper bound of differential entropy

$$h(X) \leq \log E\left[\frac{1}{p(X)}\right] = \log \int_{x\in\mathcal{X}} p(x)\frac{1}{p(x)}dx = \log|\mathcal{X}|$$

- The expression still makes sense but it is not useful usually since the sampling space can be unbounded $|\mathcal{X}| = \infty$ (for example, normally distributed X)
- Thus it makes much more sense to consider upper bound of a differential entropy constrained on the variance of the variable (why not constrained on mean?)
- It turns out that for a fixed variance σ^2, the variable will have largest differential entropy if it is normally distributed (will show later). Thus

$$h(X) \leq \log\sqrt{2\pi e\sigma^2}$$

Joint entropy

For multivariate random variable, we can extend the definition of entropy naturally as follows:

Entropy

$$H(X, Y) = E[-\log p(X, Y)]$$

and

$$H(X_1, X_2, \cdots, X_N) = E[-\log p(X_1, \cdots, X_N)]$$

Joint entropy

For multivariate random variable, we can extend the definition of entropy naturally as follows:

Entropy

$$H(X, Y) = E[-\log p(X, Y)]$$

and

$$H(X_1, X_2, \cdots, X_N) = E[-\log p(X_1, \cdots, X_N)]$$

Differential entropy

$$h(X, Y) = E[-\log p(X, Y)]$$

and

$$h(X_1, X_2, \cdots, X_N) = E[-\log p(X_1, \cdots, X_N)]$$

Conditional entropy

$$
\begin{aligned}
H(X,Y) &= E[-\log p(X,Y)] = E[-\log p(X) - \log p(Y|X)] \\
&= H(X) + \underbrace{E[-\log p(Y|X)]}_{H(Y|X)}
\end{aligned}
$$

Entropy

$$H(Y|X) \triangleq H(X,Y) - H(X)$$

Conditional entropy

$$
\begin{aligned}
H(X,Y) &= E[-\log p(X,Y)] = E[-\log p(X) - \log p(Y|X)] \\
&= H(X) + \underbrace{E[-\log p(Y|X)]}_{H(Y|X)}
\end{aligned}
$$

Entropy

$$H(Y|X) \triangleq H(X,Y) - H(X)$$

Differential entropy

$$h(Y|X) \triangleq h(X,Y) - h(X)$$

Conditional entropy

$$\begin{aligned} H(X,Y) &= E[-\log p(X,Y)] = E[-\log p(X) - \log p(Y|X)] \\ &= H(X) + \underbrace{E[-\log p(Y|X)]}_{H(Y|X)} \end{aligned}$$

Entropy

$$H(Y|X) \triangleq H(X,Y) - H(X)$$

Differential entropy

$$h(Y|X) \triangleq h(X,Y) - h(X)$$

Interpretation

Total Info. of X and Y = Info. of X + Info. of Y knowing X

Expanding conditional entropy

$$H(Y|X) = E[-\log p(Y|X)]$$

Expanding conditional entropy

$$
\begin{aligned}
H(Y|X) &= E[-\log p(Y|X)] \\
&= \sum_{x,y} -p(x,y) \log p(y|x)
\end{aligned}
$$

Expanding conditional entropy

$$
\begin{aligned}
H(Y|X) &= E[-\log p(Y|X)] \\
&= \sum_{x,y} -p(x,y)\log p(y|x) \\
&= \sum_{x} p(x) \sum_{y} -p(y|x)\log p(y|x)
\end{aligned}
$$

Expanding conditional entropy

$$
\begin{aligned}
H(Y|X) &= E[-\log p(Y|X)] \\
&= \sum_{x,y} -p(x,y)\log p(y|x) \\
&= \sum_{x} p(x) \sum_{y} -p(y|x)\log p(y|x) \\
&= \sum_{x} p(x) H(Y|x)
\end{aligned}
$$

Expanding conditional entropy

$$
\begin{aligned}
H(Y|X) &= E[-\log p(Y|X)] \\
&= \sum_{x,y} -p(x,y) \log p(y|x) \\
&= \sum_{x} p(x) \sum_{y} -p(y|x) \log p(y|x) \\
&= \sum_{x} p(x) H(Y|x)
\end{aligned}
$$

The conditional entropy $H(Y|X)$ is essentially the average of $H(Y|x)$ over all possible value of x

Chain rule

Entropy

$$H(X_1, X_2, \cdots, X_N) = H(X_1) + H(X_2|X_1) + H(X_3|X_1, X_2) + \cdots + H(X_N|X_1, X_2, \cdots, X_{N-1}).$$

Chain rule

Entropy

$$H(X_1, X_2, \cdots, X_N) = H(X_1) + H(X_2|X_1) + H(X_3|X_1, X_2) + \cdots + H(X_N|X_1, X_2, \cdots, X_{N-1}).$$

Differential entropy

$$h(X_1, X_2, \cdots, X_N) = h(X_1) + h(X_2|X_1) + h(X_3|X_1, X_2) + \cdots + h(X_N|X_1, X_2, \cdots, X_{N-1}).$$

Example

$$Pr(Rain, With\ umbrella) = 0.2 \qquad Pr(Rain, No\ umbrella) = 0.1$$
$$Pr(Sunny, With\ umbrella) = 0.2 \qquad Pr(Sunny, No\ umbrella) = 0.5$$

$$W \in \{Rain, Sunny\} \qquad U \in \{With\ umbrella, No\ umbrella\}$$

Entropies

$$H(W,U) = -0.2\log 0.2 - 0.1\log 0.1 - 0.2\log 0.2 - 0.5\log 0.5 = 1.76\ bits$$
$$H(W) = -0.3\log 0.3 - 0.7\log 0.7 = 0.88\ bits$$
$$H(U) = -0.4\log 0.4 - 0.6\log 0.6 = 0.97\ bits$$
$$H(W|U) = H(W,U) - H(U) = 0.79\ bits$$
$$H(U|W) = H(W,U) - H(W) = 0.88\ bits$$

Definition

It is often useful to gauge the difference between two distributions. KL-divergence is also known to be relative entropy. It is a way to measure the difference between two distributions. For two distributions of X, $p(x)$ and $p(y)$,

$$KL(p(x)\|q(x)) \triangleq \sum_{x\in\mathcal{X}} p(x)\log_2\frac{p(x)}{q(x)}.$$

Definition

It is often useful to gauge the difference between two distributions. KL-divergence is also known to be relative entropy. It is a way to measure the difference between two distributions. For two distributions of X, $p(x)$ and $p(y)$,

$$KL(p(x)\|q(x)) \triangleq \sum_{x\in\mathcal{X}} p(x)\log_2 \frac{p(x)}{q(x)}.$$

- N.B. If $p(x) = q(x)$ for all x, $KL(p(x)\|q(x)) = 0$ as desired

Definition

It is often useful to gauge the difference between two distributions. KL-divergence is also known to be relative entropy. It is a way to measure the difference between two distributions. For two distributions of X, $p(x)$ and $p(y)$,

$$KL(p(x)\|q(x)) \triangleq \sum_{x\in\mathcal{X}} p(x)\log_2 \frac{p(x)}{q(x)}.$$

- N.B. If $p(x)=q(x)$ for all x, $KL(p(x)\|q(x))=0$ as desired
- N.B. $KL(p(x)\|q(x)) \neq KL(q(x)\|p(x))$ in general

KL-divergence is non-negative

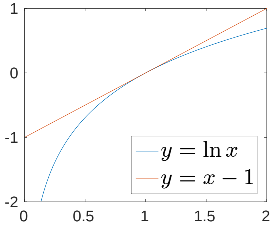
$$
\begin{aligned}
KL(p(x)\|q(x)) &= \sum_{x\in\mathcal{X}} p(x)\log_2\frac{p(x)}{q(x)} \\
&= -\sum_{x\in\mathcal{X}} p(x)\log_2\frac{q(x)}{p(x)}
\end{aligned}
$$

KL-divergence is non-negative

$$
\begin{aligned}
KL(p(x)\|q(x)) &= \sum_{x\in\mathcal{X}} p(x)\log_2 \frac{p(x)}{q(x)} \\
&= -\sum_{x\in\mathcal{X}} p(x)\log_2 \frac{q(x)}{p(x)} \\
&= -\sum_{x\in\mathcal{X}} \frac{p(x)}{\ln 2} \ln \frac{q(x)}{p(x)}
\end{aligned}
$$

KL-divergence is non-negative

$$
\begin{aligned}
KL(p(x)\|q(x)) &= \sum_{x\in\mathcal{X}} p(x)\log_2\frac{p(x)}{q(x)}\\
&= -\sum_{x\in\mathcal{X}} p(x)\log_2\frac{q(x)}{p(x)}\\
&= -\sum_{x\in\mathcal{X}} \frac{p(x)}{\ln 2}\ln\frac{q(x)}{p(x)}
\end{aligned}
$$

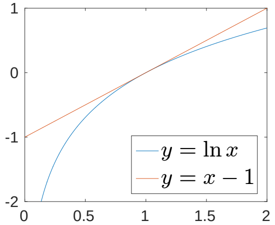


Fact

For any real x, $\ln(x) \le x - 1$. Moreover, the equality only holds when $x = 1$

KL-divergence is non-negative

$$
\begin{aligned}
KL(p(x)\|q(x)) &= \sum_{x\in\mathcal{X}} p(x)\log_2\frac{p(x)}{q(x)} \\
&= -\sum_{x\in\mathcal{X}} p(x)\log_2\frac{q(x)}{p(x)} \\
&= -\sum_{x\in\mathcal{X}} \frac{p(x)}{\ln 2}\ln\frac{q(x)}{p(x)} \\
&\geq -\sum_{x\in\mathcal{X}} \frac{p(x)}{\ln 2}\left(\frac{q(x)}{p(x)}-1\right)
\end{aligned}
$$

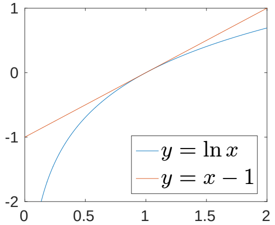


Fact

For any real x, $\ln(x)\leq x-1$. Moreover, the equality only holds when $x=1$

KL-divergence is non-negative

$$\begin{aligned}
KL(p(x)\|q(x)) &= \sum_{x\in\mathcal{X}} p(x)\log_2\frac{p(x)}{q(x)} \\
&= -\sum_{x\in\mathcal{X}} p(x)\log_2\frac{q(x)}{p(x)} \\
&= -\sum_{x\in\mathcal{X}} \frac{p(x)}{\ln 2}\ln\frac{q(x)}{p(x)} \\
&\geq -\sum_{x\in\mathcal{X}} \frac{p(x)}{\ln 2}\left(\frac{q(x)}{p(x)}-1\right) \\
&= \frac{1}{\ln 2}\left(\sum_{x\in\mathcal{X}} p(x) - \sum_{x\in\mathcal{X}} q(x)\right) = 0
\end{aligned}$$



Fact

For any real x, $\ln(x) \leq x - 1$. Moreover, the equality only holds when $x = 1$

Continuous variables

We can define KL-divergence for continuous variables in a similar manner

$$
\begin{aligned}
KL(p(x)\|q(x)) &\triangleq \int_{x\in\mathcal{X}} p(x)\log_2\frac{p(x)}{q(x)}dx \\
&= -\int_{x\in\mathcal{X}} p(x)\log_2\frac{q(x)}{p(x)}dx \\
&= -\int_{x\in\mathcal{X}} \frac{p(x)}{\ln 2}\ln\frac{q(x)}{p(x)}dx
\end{aligned}
$$

Continuous variables

We can define KL-divergence for continuous variables in a similar manner

$$
\begin{aligned}
KL(p(x)\|q(x)) &\triangleq \int_{x\in\mathcal{X}} p(x)\log_2\frac{p(x)}{q(x)}dx \\
&= -\int_{x\in\mathcal{X}} p(x)\log_2\frac{q(x)}{p(x)}dx \\
&= -\int_{x\in\mathcal{X}} \frac{p(x)}{\ln 2}\ln\frac{q(x)}{p(x)}dx \\
&\geq -\int_{x\in\mathcal{X}} \frac{p(x)}{\ln 2}\left(\frac{q(x)}{p(x)}-1\right)dx
\end{aligned}
$$

Continuous variables

We can define KL-divergence for continuous variables in a similar manner

$$
\begin{aligned}
KL(p(x)\|q(x)) &\triangleq \int_{x\in\mathcal{X}} p(x)\log_2\frac{p(x)}{q(x)}dx \\
&= -\int_{x\in\mathcal{X}} p(x)\log_2\frac{q(x)}{p(x)}dx \\
&= -\int_{x\in\mathcal{X}} \frac{p(x)}{\ln 2}\ln\frac{q(x)}{p(x)}dx \\
&\geq -\int_{x\in\mathcal{X}} \frac{p(x)}{\ln 2}\left(\frac{q(x)}{p(x)}-1\right)dx \\
&= -\frac{1}{\ln 2}\left(\int_{x\in\mathcal{X}} q(x)dx - \int_{x\in\mathcal{X}} p(x)dx\right) = 0
\end{aligned}
$$

Normal distribution has highest entropy

For fixed variance (covariance matrix), normal distribution has highest entropy

Normal distribution has highest entropy

For fixed variance (covariance matrix), normal distribution has highest entropy

Proof

Let's consider the multivariate case with a fixed covariance matrix Σ, the univariate (scalar) case is a special case thus automatically taken care of.

Normal distribution has highest entropy

For fixed variance (covariance matrix), normal distribution has highest entropy

Proof

Let's consider the multivariate case with a fixed covariance matrix Σ, the univariate (scalar) case is a special case thus automatically taken care of. Without loss of generality, let's consider zero mean. Denote $\mathcal{N}(\mathbf{x}; \mathbf{0}, \Sigma) = \phi(\mathbf{x})$.

Normal distribution has highest entropy

For fixed variance (covariance matrix), normal distribution has highest entropy

Proof

Let's consider the multivariate case with a fixed covariance matrix Σ, the univariate (scalar) case is a special case thus automatically taken care of. Without loss of generality, let's consider zero mean. Denote $\mathcal{N}(\mathbf{x};\mathbf{0},\Sigma)=\phi(\mathbf{x})$. For any other distribution $f(\mathbf{x})$ with the same covariance matrix Σ, first note that $\int_{\mathbf{x}} f(\mathbf{x})\log\phi(\mathbf{x})d\mathbf{x}=\int_{\mathbf{x}}\phi(\mathbf{x})\log\phi(\mathbf{x})d\mathbf{x}$ (to be show in the next slide).

Normal distribution has highest entropy

For fixed variance (covariance matrix), normal distribution has highest entropy

Proof

Let's consider the multivariate case with a fixed covariance matrix Σ, the univariate (scalar) case is a special case thus automatically taken care of. Without loss of generality, let's consider zero mean. Denote $\mathcal{N}(\mathbf{x}; \mathbf{0}, \Sigma) = \phi(\mathbf{x})$. For any other distribution $f(\mathbf{x})$ with the same covariance matrix Σ, first note that $\int_{\mathbf{x}} f(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x}$ (to be show in the next slide). Then,

$$0 \leq KL(f \| \phi) = \int_{\mathbf{x}} f(\mathbf{x}) \log \frac{f(\mathbf{x})}{\phi(\mathbf{x})} d\mathbf{x}$$

Normal distribution has highest entropy

For fixed variance (covariance matrix), normal distribution has highest entropy

Proof

Let's consider the multivariate case with a fixed covariance matrix Σ, the univariate (scalar) case is a special case thus automatically taken care of. Without loss of generality, let's consider zero mean. Denote $\mathcal{N}(\mathbf{x};\mathbf{0},\Sigma)=\phi(\mathbf{x})$. For any other distribution $f(\mathbf{x})$ with the same covariance matrix Σ, first note that $\int_{\mathbf{x}} f(\mathbf{x})\log\phi(\mathbf{x})d\mathbf{x}=\int_{\mathbf{x}}\phi(\mathbf{x})\log\phi(\mathbf{x})d\mathbf{x}$ (to be show in the next slide). Then,

$$0\leq KL(f\|\phi)=\int_{\mathbf{x}} f(\mathbf{x})\log\frac{f(\mathbf{x})}{\phi(\mathbf{x})}d\mathbf{x}=-h(f)-\int_{\mathbf{x}} f(\mathbf{x})\log\phi(\mathbf{x})d\mathbf{x}$$

Normal distribution has highest entropy

For fixed variance (covariance matrix), normal distribution has highest entropy

Proof

Let's consider the multivariate case with a fixed covariance matrix Σ, the univariate (scalar) case is a special case thus automatically taken care of. Without loss of generality, let's consider zero mean. Denote $\mathcal{N}(\mathbf{x};\mathbf{0},\Sigma)=\phi(\mathbf{x})$. For any other distribution $f(\mathbf{x})$ with the same covariance matrix Σ, first note that $\int_{\mathbf{x}} f(\mathbf{x})\log\phi(\mathbf{x})d\mathbf{x}=\int_{\mathbf{x}}\phi(\mathbf{x})\log\phi(\mathbf{x})d\mathbf{x}$ (to be show in the next slide). Then,

$$
\begin{aligned}
0 \leq & KL(f\|\phi)=\int_{\mathbf{x}} f(\mathbf{x})\log\frac{f(\mathbf{x})}{\phi(\mathbf{x})}d\mathbf{x}=-h(f)-\int_{\mathbf{x}} f(\mathbf{x})\log\phi(\mathbf{x})d\mathbf{x}\\
=& -h(f)-\int_{\mathbf{x}}\phi(\mathbf{x})\log\phi(\mathbf{x})d\mathbf{x}=-h(f)+h(\phi)
\end{aligned}
$$

$\int_{\mathbf{x}} f(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x}$

$$\int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \phi(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2}\mathbf{x}^T \Sigma^{-1} \mathbf{x} \right] d\mathbf{x}$$

$\int_{\mathbf{x}} f(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x}$

$$\int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \phi(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2}\mathbf{x}^T \Sigma^{-1} \mathbf{x} \right] d\mathbf{x}$$

$$= \int_{x} \phi(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2} \sum_{i,j} x_i \left[\Sigma^{-1}\right]_{i,j} x_j \right] d\mathbf{x}$$

$\int_{\mathbf{x}} f(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x}$

$$\int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \phi(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2}\mathbf{x}^T \Sigma^{-1} \mathbf{x} \right] d\mathbf{x}$$

$$= \int_{x} \phi(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2} \sum_{i,j} x_i \left[\Sigma^{-1}\right]_{i,j} x_j \right] d\mathbf{x}$$

$$= \int_{\mathbf{x}} \phi(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2} \sum_{i,j} \left[\Sigma^{-1}\right]_{i,j} x_i x_j \right] d\mathbf{x}$$

$\int_{\mathbf{x}} f(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x}$

$$\int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \phi(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2}\mathbf{x}^T \Sigma^{-1} \mathbf{x} \right] d\mathbf{x}$$

$$= \int_{x} \phi(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2} \sum_{i,j} x_i \left[\Sigma^{-1}\right]_{i,j} x_j \right] d\mathbf{x}$$

$$= \int_{\mathbf{x}} \phi(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2} \sum_{i,j} \left[\Sigma^{-1}\right]_{i,j} x_i x_j \right] d\mathbf{x}$$

$$= \int_{\mathbf{x}} f(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2} \sum_{i,j} \left[\Sigma^{-1}\right]_{i,j} x_i x_j \right] d\mathbf{x}$$

$\int_{\mathbf{x}} f(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x}$

$$
\begin{aligned}
\int_{\mathbf{x}} \phi(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x} &= \int_{\mathbf{x}} \phi(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2}\mathbf{x}^T \Sigma^{-1} \mathbf{x} \right] d\mathbf{x} \\
&= \int_{x} \phi(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2} \sum_{i,j} x_i \left[\Sigma^{-1}\right]_{i,j} x_j \right] d\mathbf{x} \\
&= \int_{\mathbf{x}} \phi(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2} \sum_{i,j} \left[\Sigma^{-1}\right]_{i,j} x_i x_j \right] d\mathbf{x} \\
&= \int_{\mathbf{x}} f(\mathbf{x}) \left[-\log \sqrt{\det(2\pi\Sigma)} - \frac{1}{2} \sum_{i,j} \left[\Sigma^{-1}\right]_{i,j} x_i x_j \right] d\mathbf{x} \\
&= \int_{\mathbf{x}} f(\mathbf{x}) \log \phi(\mathbf{x}) d\mathbf{x}
\end{aligned}
$$

Application: Thiel index

- Measure economic inequality among different groups or for a group of individuals
- Let p_i be the economic wealth proportion of group i, and q_i be the population size proportion of group i
- Thiel index is simply $KL(p||q)$
- Let's apply to a group of N individuals.
    - If they all have the same wealth, both p and q are uniform ($p_i = q_i = 1/N$), thus Thiel index $= KL(p||q) = 0$
    - If one of them own everything, q is uniform but p is a δ-function. Thus Thiel index $= KL(p||q) = \sum_i p_i \log \frac{p_i}{q_i} = \log \frac{1}{1/N} = \log N$

Application: Cross-entropy and cross-entropy error

In machine learning, it is often needed to assess the quality of a trained system. Consider the example of classifying an the political affliation of an individual

```
computed       | targets              | correct?
-----------------------------------------------
0.3  0.3  0.4  | 0  0  1 (democrat)   | yes
0.3  0.4  0.3  | 0  1  0 (republican) | yes
0.1  0.2  0.7  | 1  0  0 (other)      | no
```

```
computed       | targets              | correct?
-----------------------------------------------
0.1  0.2  0.7  | 0  0  1 (democrat)   | yes
0.1  0.7  0.2  | 0  1  0 (republican) | yes
0.3  0.4  0.3  | 1  0  0 (other)      | no
```

In a first glance, both examples appear to work equally well (or bad). Both have one classification error. However, a closer look will suggest the prediction of LHS is worse than RHS (why?)

(https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/)

Application: Cross-entropy and cross-entropy error

In machine learning, it is often needed to assess the quality of a trained system. Consider the example of classifying an the political affliation of an individual

```
computed        | targets              | correct?
----------------------------------------------
0.3  0.3  0.4   | 0  0  1 (democrat)   | yes
0.3  0.4  0.3   | 0  1  0 (republican) | yes
0.1  0.2  0.7   | 1  0  0 (other)      | no
```

```
computed        | targets              | correct?
----------------------------------------------
0.1  0.2  0.7   | 0  0  1 (democrat)   | yes
0.1  0.7  0.2   | 0  1  0 (republican) | yes
0.3  0.4  0.3   | 1  0  0 (other)      | no
```

In a first glance, both examples appear to work equally well (or bad). Both have one classification error. However, a closer look will suggest the prediction of LHS is worse than RHS (why?) For a better assessment, we can treat both the computed result and the target result as distribution and compare them with KL-divergence. Namely

$$KL(p_{target}\|p_{computed}) = \sum_{group} p_{target}(group) \log \frac{p_{target}(group)}{p_{computed}(group)}$$

$$= -H(p_{target}) \underbrace{- \sum_{group} p_{target}(group) \log p_{computed}(group)}_{cross\ \ entropy}$$

(https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/)

Application: Cross-entropy and cross-entropy error

$$
\begin{aligned}
Cross\ entropy(p\|q) &\triangleq \sum_x p(x)\log\frac{1}{q(x)} = E_p[-\log q(X)] \\
&= H(p) + KL(p\|q)
\end{aligned}
$$

Application: Cross-entropy and cross-entropy error

$$\begin{aligned} Cross\ entropy(p\|q) &\triangleq \sum_x p(x)\log\frac{1}{q(x)} = E_p[-\log q(X)] \\ &= H(p) + KL(p\|q) \end{aligned}$$

- To compute KL-divergence, one needs to find $H(p_{target})$, which is independent of the machine learning system and thus does not reflect the performance of the system

Application: Cross-entropy and cross-entropy error

$$
\begin{aligned}
Cross\ entropy(p\|q) &\triangleq \sum_x p(x)\log\frac{1}{q(x)} = E_p[-\log q(X)] \\
&= H(p) + KL(p\|q)
\end{aligned}
$$

- To compute KL-divergence, one needs to find $H(p_{target})$, which is independent of the machine learning system and thus does not reflect the performance of the system
- Thus in practice, cross-entropy is commonly used instead of KL-divergence to measure the performance of a machine learning system

Example: Text processing

- In text processing, it is common that one may need to measure the similiarity between two documents D_1 and D_2.

Example: Text processing

- In text processing, it is common that one may need to measure the similiarity between two documents D_1 and D_2.
- How to represent documents? One may use the "bag of words". That is, to convert document into a vector of numbers. Each number is the count of a corresponding word

Example: Text processing

- In text processing, it is common that one may need to measure the similiarity between two documents D_1 and D_2.
- How to represent documents? One may use the "bag of words". That is, to convert document into a vector of numbers. Each number is the count of a corresponding word
- One can then compares two documents using cross entropy

$$Cross\ entropy(p_1 \| p_2) = \sum_w p_1(w) \log \frac{1}{p_2(w)},$$

where p_1 and p_2 are the word distributions of documents D_1 and D_2, respectively

TF-IDF and cross entropy

It may be also interesting of comparing word distribution of a document to the word distribution across all documents That is, let q be the word distribution across all documents,

$$\begin{aligned} Cross\ entropy(p_1\|q) &= \sum_w p_1(w)\log\frac{1}{q(w)} \\ &= \sum_w \underbrace{\frac{\#\ w \text{ in } D_1}{\text{total } \#\text{ words in } D_1}\log\frac{\text{total } \#\text{ docs}}{\#\text{ doc with } w}}_{TF\text{-}IDF(w)}, \end{aligned}$$

where $TF\text{-}IDF(w)$, short for term frequency-inverse document frequency, can reflect how important of the word w to the target document and can be used in search engine

Definition

As $H(X)$ is equivalent to the information revealed by X and $H(X|Y)$ the remaining information of X knowing Y, we expect that $H(X) - H(X|Y)$ is the information of X shared by $Y \Rightarrow$ "mutual information"

$$I(X;Y) \triangleq H(X) - H(X|Y)$$

Definition

As $H(X)$ is equivalent to the information revealed by X and $H(X|Y)$ the remaining information of X knowing Y, we expect that $H(X) - H(X|Y)$ is the information of X shared by $Y \Rightarrow$ "mutual information"

$$I(X;Y) \triangleq H(X) - H(X|Y)$$

Similarly, we can define the "conditional mutual information" shared between X and Y given Z as

$$I(X;Y|Z) \triangleq H(X|Z) - H(X|Y,Z)$$

Property of mutual information

$I(X;Y) = I(Y;X) \geq 0$

The definition is symmetric and non-negative as desired.

$$I(X;Y) = H(X) - H(X|Y) = E[-\log p(X)] - E[-\log p(X|Y)]$$

Property of mutual information

$I(X;Y) = I(Y;X) \geq 0$

The definition is symmetric and non-negative as desired.

$$
\begin{aligned}
I(X;Y) =& H(X) - H(X|Y) = E[-\log p(X)] - E[-\log p(X|Y)] \\
=& -\sum_{x} p(x) \log p(x) + \sum_{x,y} p(x,y) \log p(x|y)
\end{aligned}
$$

Property of mutual information

$I(X;Y) = I(Y;X) \geq 0$

The definition is symmetric and non-negative as desired.

$$
\begin{aligned}
I(X;Y) =& H(X) - H(X|Y) = E[-\log p(X)] - E[-\log p(X|Y)] \\
&= -\sum_x p(x)\log p(x) + \sum_{x,y} p(x,y)\log p(x|y) \\
&= -\sum_{x,y} p(x,y)\log p(x) + \sum_{x,y} p(x,y)\log p(x|y) = \sum_{x,y} p(x,y)\log\frac{p(x|y)}{p(x)}
\end{aligned}
$$

Property of mutual information

$I(X;Y) = I(Y;X) \geq 0$

The definition is symmetric and non-negative as desired.

$$
\begin{aligned}
I(X;Y) =& H(X) - H(X|Y) = E[-\log p(X)] - E[-\log p(X|Y)] \\
=& -\sum_{x} p(x)\log p(x) + \sum_{x,y} p(x,y)\log p(x|y) \\
=& -\sum_{x,y} p(x,y)\log p(x) + \sum_{x,y} p(x,y)\log p(x|y) = \sum_{x,y} p(x,y)\log\frac{p(x|y)}{p(x)} \\
=& \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}
\end{aligned}
$$

Property of mutual information

$I(X;Y) = I(Y;X) \geq 0$

The definition is symmetric and non-negative as desired.

$$
\begin{aligned}
I(X;Y) =& H(X) - H(X|Y) = E[-\log p(X)] - E[-\log p(X|Y)] \\
&= -\sum_x p(x)\log p(x) + \sum_{x,y} p(x,y)\log p(x|y) \\
&= -\sum_{x,y} p(x,y)\log p(x) + \sum_{x,y} p(x,y)\log p(x|y) = \sum_{x,y} p(x,y)\log\frac{p(x|y)}{p(x)} \\
&= \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} = KL(p(x,y)\|p(x)p(y)) \geq 0
\end{aligned}
$$

Property of conditional mutual information

$I(X;Y|Z) = I(Y;X|Z) \geq 0$

The definition is symmetric and non-negative as desired.

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z) = E[-\log p(X|Z)] - E[-\log p(X|Y,Z)]$$

Property of conditional mutual information

$I(X;Y|Z) = I(Y;X|Z) \geq 0$

The definition is symmetric and non-negative as desired.

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z) = E[-\log p(X|Z)] - E[-\log p(X|Y,Z)]$$
$$= -\sum_{x,z} p(x,z) \log p(x|z) + \sum_{x,y,z} p(x,y,z) \log p(x|y,z)$$

Property of conditional mutual information

$I(X;Y|Z) = I(Y;X|Z) \geq 0$

The definition is symmetric and non-negative as desired.

$$
\begin{aligned}
I(X;Y|Z) =& H(X|Z) - H(X|Y,Z) = E[-\log p(X|Z)] - E[-\log p(X|Y,Z)] \\
&= -\sum_{x,z} p(x,z) \log p(x|z) + \sum_{x,y,z} p(x,y,z) \log p(x|y,z) \\
&= -\sum_{x,y,z} p(x,y,z) \log p(x|z) + \sum_{x,y,z} p(x,y,z) \log p(x|y,z) \\
&= \sum_{x,y,z} p(x,y,z) \log \frac{p(x|y,z)}{p(x|z)}
\end{aligned}
$$

Property of conditional mutual information

$I(X;Y|Z) = I(Y;X|Z) \geq 0$

The definition is symmetric and non-negative as desired.

$$
\begin{aligned}
I(X;Y|Z) =& H(X|Z) - H(X|Y,Z) = E[-\log p(X|Z)] - E[-\log p(X|Y,Z)] \\
&= -\sum_{x,z} p(x,z)\log p(x|z) + \sum_{x,y,z} p(x,y,z)\log p(x|y,z) \\
&= -\sum_{x,y,z} p(x,y,z)\log p(x|z) + \sum_{x,y,z} p(x,y,z)\log p(x|y,z) \\
&= \sum_{x,y,z} p(x,y,z)\log \frac{p(x|y,z)}{p(x|z)} \\
&= \sum_{z} p(z) \sum_{x,y} p(x,y|z)\log \frac{p(x,y|z)}{p(x|z)p(y|z)} \\
&= \sum_{z} p(z) KL(p(x,y|z) \| p(x|z)p(y|z)) \geq 0
\end{aligned}
$$

Independence and mutual information

$I(X;Y)=0 \Leftrightarrow X \perp Y$

$$I(X;Y)=KL(p(x,y)\|p(x)p(y))=0$$

implies $p(x,y)=p(x)p(y)$. Therefore $X \perp Y$

Independence and mutual information

$I(X;Y)=0 \Leftrightarrow X \perp Y$

$$I(X;Y)=KL(p(x,y)\|p(x)p(y))=0$$

implies $p(x,y)=p(x)p(y)$. Therefore $X \perp Y$

$I(X;Y|Z)=0 \Leftrightarrow X \perp Y|Z$

$$I(X;Y|Z)=\sum_z p(z)KL(p(x,y|z)\|p(x|z)p(y|z))=0$$

implies $p(x,y|z)=p(x|z)p(y|z)$ for all z s.t. $p(z)>0$. Therefore $X \perp Y|Z$

Remark

This is just as what we expect. If there is no share information between X and Y, they should be indepedent!

Chain rule for mutual information

$$I(X_1, X_2, \cdots, X_N | Y)$$

Chain rule for mutual information

$$
\begin{aligned}
&I(X_1, X_2, \cdots, X_N|Y) \\
=&H(X_1, X_2, \cdots, X_N) - H(X_1, X_2, \cdots, X_N|Y)
\end{aligned}
$$

Chain rule for mutual information

$$
\begin{aligned}
& I(X_1, X_2, \cdots, X_N|Y) \\
=& H(X_1, X_2, \cdots, X_N) - H(X_1, X_2, \cdots, X_N|Y) \\
=& \sum_{i=1}^{N} H(X_i|X^{i-1}) - H(X_i|X^{i-1}, Y)
\end{aligned}
$$

N.B. $X^N = X_1, X_2, \cdots, X_N$

Chain rule for mutual information

$$
\begin{aligned}
& I(X_1, X_2, \cdots, X_N|Y) \\
=& H(X_1, X_2, \cdots, X_N) - H(X_1, X_2, \cdots, X_N|Y) \\
=& \sum_{i=1}^{N} H(X_i|X^{i-1}) - H(X_i|X^{i-1}, Y) \\
=& \sum_{i=1}^{N} I(X_i; Y|X^{i-1})
\end{aligned}
$$

N.B. $X^N = X_1, X_2, \cdots, X_N$

Mutual information for continuous variables

For continuous X, Y, Z, we can define $I(X;Y) = h(X) - h(X|Y)$ and
$I(X;Y|Z) = h(X|Z) - h(X|Y,Z)$
Then, the followings still hold true

Mutual information for continuous variables

For continuous X, Y, Z, we can define $I(X;Y) = h(X) - h(X|Y)$ and $I(X;Y|Z) = h(X|Z) - h(X|Y,Z)$
Then, the followings still hold true

- $I(X;Y) = KL(p(x,y)\|p(x)p(y)) = I(Y;X) \geq 0$

Mutual information for continuous variables

For continuous X, Y, Z, we can define $I(X;Y) = h(X) - h(X|Y)$ and $I(X;Y|Z) = h(X|Z) - h(X|Y,Z)$
Then, the followings still hold true

- $I(X;Y) = KL(p(x,y)\|p(x)p(y)) = I(Y;X) \geq 0$
- $I(X;Y|Z) = \int_z p(z)KL(p(x,y|z)\|p(x|z)p(y|z))dz = I(Y;X|Z) \geq 0$

Mutual information for continuous variables

For continuous X, Y, Z, we can define $I(X;Y) = h(X) - h(X|Y)$ and $I(X;Y|Z) = h(X|Z) - h(X|Y,Z)$
Then, the followings still hold true

- $I(X;Y) = KL(p(x,y)\|p(x)p(y)) = I(Y;X) \geq 0$
- $I(X;Y|Z) = \int_z p(z)KL(p(x,y|z)\|p(x|z)p(y|z))dz = I(Y;X|Z) \geq 0$
- $I(X;Y) = 0 \Leftrightarrow X \perp Y$

Mutual information for continuous variables

For continuous X, Y, Z, we can define $I(X;Y) = h(X) - h(X|Y)$ and $I(X;Y|Z) = h(X|Z) - h(X|Y,Z)$
Then, the followings still hold true

- $I(X;Y) = KL(p(x,y)\|p(x)p(y)) = I(Y;X) \geq 0$
- $I(X;Y|Z) = \int_z p(z)KL(p(x,y|z)\|p(x|z)p(y|z))dz = I(Y;X|Z) \geq 0$
- $I(X;Y) = 0 \Leftrightarrow X \perp Y$
- $I(X;Y|Z) = 0 \Leftrightarrow X \perp Y|Z$

Mutual information for continuous variables

For continuous X, Y, Z, we can define $I(X;Y) = h(X) - h(X|Y)$ and $I(X;Y|Z) = h(X|Z) - h(X|Y,Z)$

Then, the followings still hold true

- $I(X;Y) = KL(p(x,y)\|p(x)p(y)) = I(Y;X) \geq 0$
- $I(X;Y|Z) = \int_z p(z)KL(p(x,y|z)\|p(x|z)p(y|z))dz = I(Y;X|Z) \geq 0$
- $I(X;Y) = 0 \Leftrightarrow X \perp Y$
- $I(X;Y|Z) = 0 \Leftrightarrow X \perp Y|Z$
- $I(X_1, X_2, \cdots, X_N|Y) = \sum_{i=1}^{N} I(X_i; Y|X^{i-1})$

Conditioning reduces entropy

Given more information, the residual information (uncertainty) should decrease.

Conditioning reduces entropy

Given more information, the residual information (uncertainty) should decrease. More precisely,

$$H(X) \geq H(X|Y) \qquad H(X|Y) \geq H(X|Y,Z)$$

This is obvious from our previous discussion since
$H(X) - H(X|Y) = I(X;Y) \geq 0$ and
$H(X|Y) - H(X|Y,Z) = I(X;Z|Y) \geq 0$

Conditioning reduces entropy

Given more information, the residual information (uncertainty) should decrease. More precisely,

$$H(X) \geq H(X|Y) \qquad H(X|Y) \geq H(X|Y,Z)$$

This is obvious from our previous discussion since
$H(X) - H(X|Y) = I(X;Y) \geq 0$ and
$H(X|Y) - H(X|Y,Z) = I(X;Z|Y) \geq 0$

Of course, we also have

$$h(X) \geq h(X|Y) \qquad h(X|Y) \geq h(X|Y,Z)$$

since $h(X) - h(X|Y) = I(X;Y) \geq 0$ and
$h(X|Y) - h(X|Y) = I(X;Z|Y) \geq 0$

Data processing inequality

If random variables X, Y, Z satisfy $X \leftrightarrow Y \leftrightarrow Z$, then

$$I(X;Y) \geq I(X;Z).$$

Proof

$$I(X;Y) = I(X;Y,Z) - I(X;Z|Y)$$

Data processing inequality

If random variables X, Y, Z satisfy $X \leftrightarrow Y \leftrightarrow Z$, then

$$I(X;Y) \geq I(X;Z).$$

Proof

$$\begin{aligned} I(X;Y) &= I(X;Y,Z) - I(X;Z|Y) \\ &= I(X;Y,Z) \qquad\qquad (\text{since } X \leftrightarrow Y \leftrightarrow Z) \end{aligned}$$

Data processing inequality

If random variables X, Y, Z satisfy $X \leftrightarrow Y \leftrightarrow Z$, then

$$I(X;Y) \geq I(X;Z).$$

Proof

$$\begin{aligned} I(X;Y) &= I(X;Y,Z) - I(X;Z|Y) \\ &= I(X;Y,Z) \qquad \text{(since } X \leftrightarrow Y \leftrightarrow Z\text{)} \\ &= I(X;Z) + I(X;Y|Z) \\ &\geq I(X;Z) \end{aligned}$$

Application: perfect secrecy

Example (A simple cryptography example)

- Say you have a very personal letter that you don't want to let anyone else except some special someone to read

Application: perfect secrecy

Example (A simple cryptography example)

- Say you have a very personal letter that you don't want to let anyone else except some special someone to read
- You will first encrypt the letter to some code. To decrypt the message, you will need some key and you will also pass it to your special someone.

Application: perfect secrecy

Example (A simple cryptography example)

- Say you have a very personal letter that you don't want to let anyone else except some special someone to read
- You will first encrypt the letter to some code. To decrypt the message, you will need some key and you will also pass it to your special someone. Translate to the cryptography language/symbols
  - Letter: plaintext message M
  - Code: ciphertext C
  - Key: key K

Application: perfect secrecy

Example (A simple cryptography example)

- Say you have a very personal letter that you don't want to let anyone else except some special someone to read
- You will first encrypt the letter to some code. To decrypt the message, you will need some key and you will also pass it to your special someone. Translate to the cryptography language/symbols
  - Letter: plaintext message M
  - Code: ciphertext C
  - Key: key K

Remark

Shannon's result: to ensure perfect secrecy, we can show that $H(M) \leq H(K)$

Application: perfect secrecy

Recall that M, C, K be plaintext message, ciphertext, and key, respectively

Assumption

We will assume here that we have a ***non-probabilistic*** *encryption scheme. In other words, each plaintext message maps to a unique ciphertext given a fixed key. So there is no ambiguity during decoding. Therefore,* $H(M|C, K) = 0$

Application: perfect secrecy

Recall that M, C, K be plaintext message, ciphertext, and key, respectively

Assumption

*We will assume here that we have a **non-probabilistic** encryption scheme. In other words, each plaintext message maps to a unique ciphertext given a fixed key. So there is no ambiguity during decoding. Therefore,* $H(M|C, K) = 0$

Remark (Independence)

For perfect secrecy, one should not be able to deduce anything regarding the message from the ciphertext. Therefore, C and M should be independent.

Application: perfect secrecy

Recall that M, C, K be plaintext message, ciphertext, and key, respectively

Assumption

We will assume here that we have a ***non-probabilistic*** *encryption scheme. In other words, each plaintext message maps to a unique ciphertext given a fixed key. So there is no ambiguity during decoding. Therefore,* $H(M|C,K) = 0$

Remark (Independence)

For perfect secrecy, one should not be able to deduce anything regarding the message from the ciphertext. Therefore, C and M should be independent. Thus,
$I(C;M) = 0 \Rightarrow H(M) = H(M|C) + I(C;M) = H(M|C)$

Application: perfect secrecy

Lemma (Entropy bound)

For any ***non-probabilistic*** *encryption scheme,* $H(M|C) \leq H(K|C)$

Application: perfect secrecy

Lemma (Entropy bound)

For any ***non-probabilistic*** *encryption scheme,* $H(M|C) \leq H(K|C)$

Proof.

Recall that for non-probabilistic encryption scheme, $H(M|K, C) = 0 \Rightarrow$ $H(M|C) \leq H(M, K|C)$

Application: perfect secrecy

Lemma (Entropy bound)

*For any **non-probabilistic** encryption scheme,* $H(M|C) \leq H(K|C)$

Proof.

Recall that for non-probabilistic encryption scheme, $H(M|K,C) = 0 \Rightarrow$
$H(M|C) \leq H(M,K|C) = H(K|C) + H(M|K,C) = H(K|C)$ □

Application: perfect secrecy

Lemma (Entropy bound)

*For any **non-probabilistic** encryption scheme,* $H(M|C) \leq H(K|C)$

Proof.

Recall that for non-probabilistic encryption scheme, $H(M|K,C) = 0 \Rightarrow$
$H(M|C) \leq H(M,K|C) = H(K|C) + H(M|K,C) = H(K|C)$ $\square$

Corollary (Entropy bound)

For any non-probabilistic encryption scheme, $H(M|C) \leq H(K)$

Application: perfect secrecy

Lemma (Entropy bound)

*For any **non-probabilistic** encryption scheme, $H(M|C) \le H(K|C)$*

Proof.

Recall that for non-probabilistic encryption scheme, $H(M|K,C) = 0 \Rightarrow H(M|C) \le H(M,K|C) = H(K|C) + H(M|K,C) = H(K|C)$ $\square$

Corollary (Entropy bound)

For any non-probabilistic encryption scheme, $H(M|C) \le H(K)$

Theorem (Perfect secrecy)

We have perfect secrecy if $H(M) \le H(K)$

Application: perfect secrecy

Lemma (Entropy bound)

*For any **non-probabilistic** encryption scheme,* $H(M|C) \leq H(K|C)$

Proof.

Recall that for non-probabilistic encryption scheme, $H(M|K,C) = 0 \Rightarrow$
$H(M|C) \leq H(M,K|C) = H(K|C) + H(M|K,C) = H(K|C)$ □

Corollary (Entropy bound)

For any non-probabilistic encryption scheme, $H(M|C) \leq H(K)$

Theorem (Perfect secrecy)

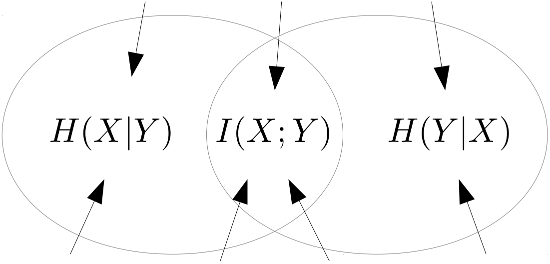
We have perfect secrecy if $H(M) \leq H(K)$

Proof.

Combine Corollary (Entropy bound) and Remark (Independence) □

Summary

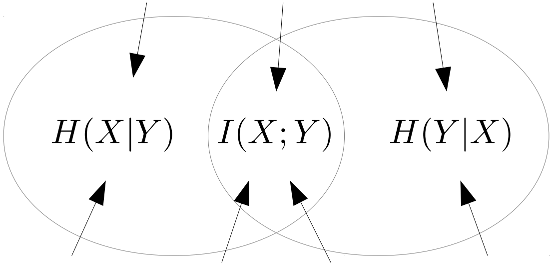
$$H(X,Y) = H(X|Y) + I(X;Y) + H(Y|X)$$



$$H(X) = H(X|Y) + I(X;Y) \qquad I(X;Y) + H(Y|X) = H(Y)$$

Review

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



$$H(X) = H(X|Y) + I(X;Y) \qquad I(X;Y) + H(Y|X) = H(Y)$$

Review

- Conditioning reduces entropy

Review

- Conditioning reduces entropy
- Chain rules:
  - $H(X, Y, Z)$

Review

- Conditioning reduces entropy
- Chain rules:
  - $H(X,Y,Z) = H(Z) \; + H(Y|X) \; + H(Z|X,Y)$
  - $H(X,Y,U|V)$

Review

- Conditioning reduces entropy
- Chain rules:
  - $H(X,Y,Z) = H(Z) \; + H(Y|X) \; + H(Z|X,Y)$
  - $H(X,Y,U|V) = H(X|V) + H(Y|X,V) + H(U|Y,X,V)$
  - $I(X,Y,Z;U)$

Review

- Conditioning reduces entropy
- Chain rules:
  - $H(X,Y,Z) = H(Z) + H(Y|X) + H(Z|X,Y)$
  - $H(X,Y,U|V) = H(X|V) + H(Y|X,V) + H(U|Y,X,V)$
  - $I(X,Y,Z;U) = I(X;U) + I(Y;U|X) + I(Z;U|X,Y)$
  - $I(X,Y,Z;U|V)$

Review

- Conditioning reduces entropy
- Chain rules:
  - $H(X, Y, Z) = H(Z) + H(Y|X) + H(Z|X, Y)$
  - $H(X, Y, U|V) = H(X|V) + H(Y|X, V) + H(U|Y, X, V)$
  - $I(X, Y, Z; U) = I(X; U) + I(Y; U|X) + I(Z; U|X, Y)$
  - $I(X, Y, Z; U|V) = I(X; U|V) + I(Y; U|V, X) + I(Z; U|V, X, Y)$
- Data processing inequality: if $X \perp Y|Z$,

Review

- Conditioning reduces entropy
- Chain rules:
  - $H(X,Y,Z) = H(Z) \ + H(Y|X) \ + H(Z|X,Y)$
  - $H(X,Y,U|V) = H(X|V) + H(Y|X,V) + H(U|Y,X,V)$
  - $I(X,Y,Z;U) = I(X;U) + I(Y;U|X) + I(Z;U|X,Y)$
  - $I(X,Y,Z;U|V) = I(X;U|V) + I(Y;U|V,X) + \ I(Z;U|V,X,Y)$
- Data processing inequality: if $X \perp Y|Z$, $I(X;Y) \geq I(X;Z)$
- Independence and mutual information:
  - $X \perp Y \Leftrightarrow$

Review

- Conditioning reduces entropy
- Chain rules:
  - $H(X,Y,Z) = H(Z) \; + H(Y|X) \; + H(Z|X,Y)$
  - $H(X,Y,U|V) = H(X|V) + H(Y|X,V) + H(U|Y,X,V)$
  - $I(X,Y,Z;U) = I(X;U) + I(Y;U|X) + I(Z;U|X,Y)$
  - $I(X,Y,Z;U|V) = I(X;U|V) + I(Y;U|V,X) + \; I(Z;U|V,X,Y)$
- Data processing inequality: if $X \perp Y|Z$, $I(X;Y) \geq I(X;Z)$
- Independence and mutual information:
  - $X \perp Y \Leftrightarrow I(X;Y) = 0$
  - $X \perp Y|Z \Leftrightarrow$

Review

- Conditioning reduces entropy
- Chain rules:
  - $H(X,Y,Z) = H(Z) \;+ H(Y|X) \;+ H(Z|X,Y)$
  - $H(X,Y,U|V) = H(X|V) + H(Y|X,V) + H(U|Y,X,V)$
  - $I(X,Y,Z;U) = I(X;U) + I(Y;U|X) + I(Z;U|X,Y)$
  - $I(X,Y,Z;U|V) = I(X;U|V) + I(Y;U|V,X) + \; I(Z;U|V,X,Y)$
- Data processing inequality: if $X \perp Y|Z$, $I(X;Y) \geq I(X;Z)$
- Independence and mutual information:
  - $X \perp Y \Leftrightarrow I(X;Y) = 0$
  - $X \perp Y|Z \Leftrightarrow I(X;Y|Z) = 0$
- KL-divergence: $KL(p||q) \triangleq$

Review

- Conditioning reduces entropy
- Chain rules:
  - $H(X,Y,Z) = H(Z) + H(Y|X) + H(Z|X,Y)$
  - $H(X,Y,U|V) = H(X|V) + H(Y|X,V) + H(U|Y,X,V)$
  - $I(X,Y,Z;U) = I(X;U) + I(Y;U|X) + I(Z;U|X,Y)$
  - $I(X,Y,Z;U|V) = I(X;U|V) + I(Y;U|V,X) + I(Z;U|V,X,Y)$
- Data processing inequality: if $X \perp Y|Z$, $I(X;Y) \geq I(X;Z)$
- Independence and mutual information:
  - $X \perp Y \Leftrightarrow I(X;Y) = 0$
  - $X \perp Y|Z \Leftrightarrow I(X;Y|Z) = 0$
- KL-divergence: $KL(p||q) \triangleq \sum_x p(x) \log \frac{p(x)}{q(x)} \geq 0$

This time

- Identification/Decision trees
- Random forests
- Law of Large Number
- Asymptotic equipartition (AEP) and typical sequences

Vampire database

Romanian Data Base

Vampire?	Shadow?	Garlic?	Complexion?	Accent?
No	?	Yes	Pale	None
No	Yes	Yes	Ruddy	None
Yes	?	No	Ruddy	None
Yes	No	No	Average	Heavy
Yes	?	No	Average	Odd
No	Yes	No	Pale	Heavy
No	Yes	No	Average	Heavy
No	?	Yes	Ruddy	Odd

(https://www.youtube.com/watch?v=SXBG3RGr_Rc)

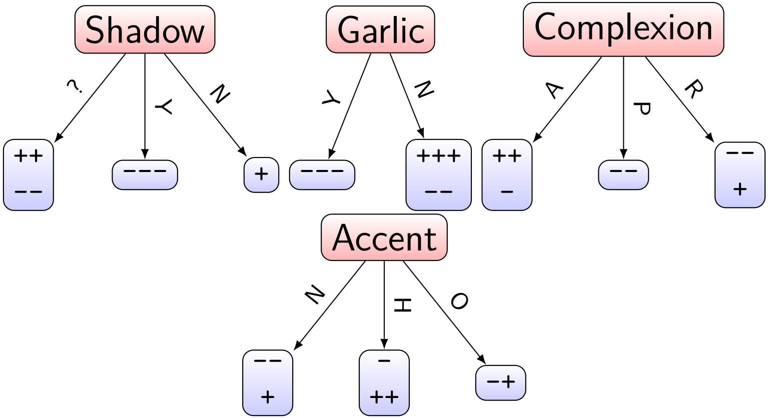
Identifying vampire

Goal: Design a set of tests to identify vampires

Potential difficulties

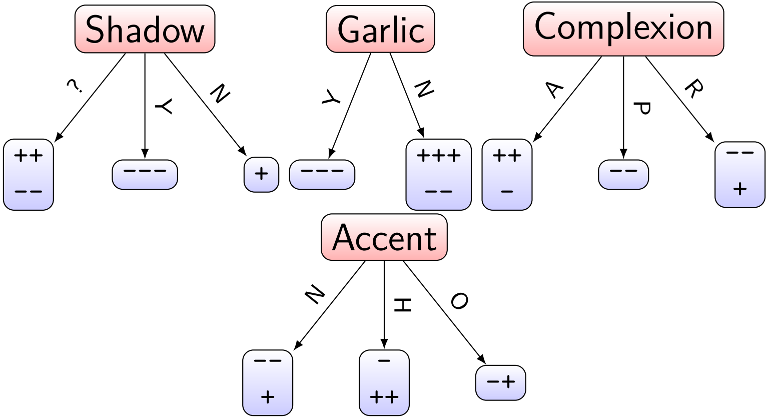
- Non-numerical data
- Some information may not matter
- Some may matter only sometimes
- Tests may be costly $\Rightarrow$ conduct as few as possible

Test trees



$+$: Vampire $\quad -$: Not vampire

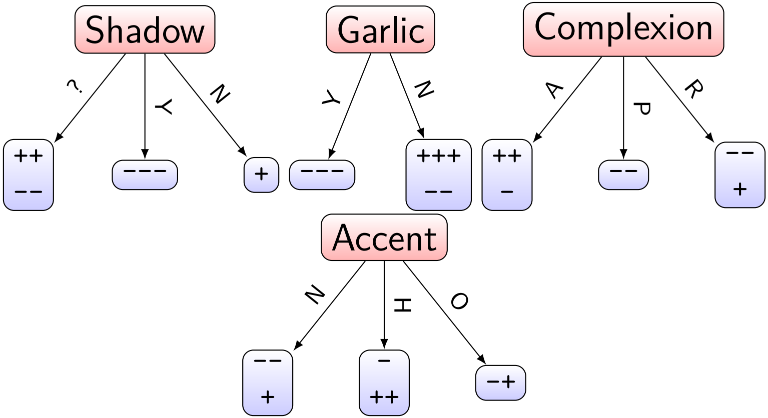
Test trees



+ : Vampire − : Not vampire

How to pick a good test?

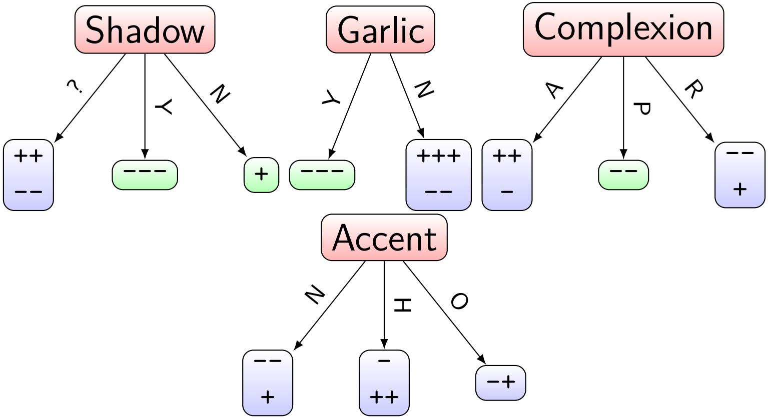
Test trees



+ : Vampire − : Not vampire

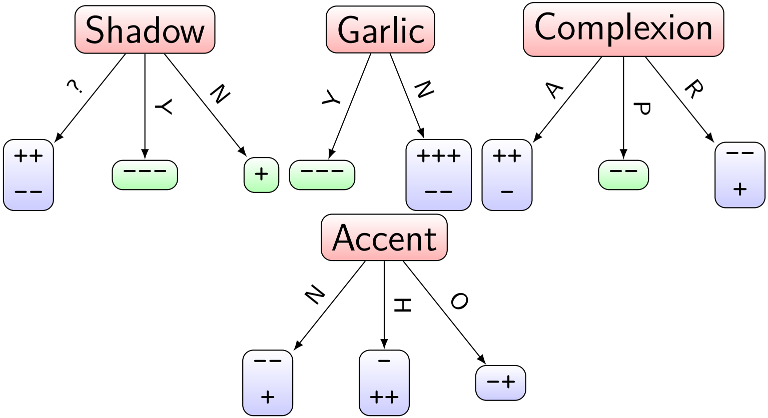
How to pick a good test? Pick test that identifies most vampires (and non-vampires)!

Sizes of homogeneous sets



$+$: Vampire $\quad -$: Not vampire

Sizes of homogeneous sets

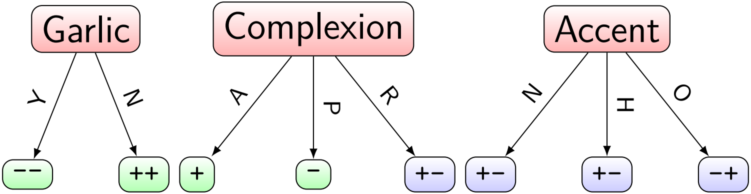


$+$: Vampire $-$: Not vampire

Shadow: 4 Garlic: 3 Complexion: 2 Accent: 0

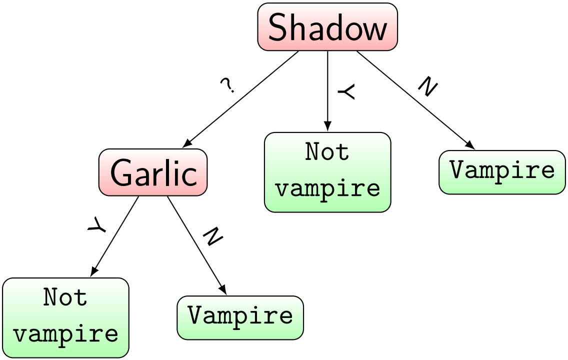
Picking second test

Let say we pick "shadow" as the first test after all. Then, for the remaining unclassified individuals,

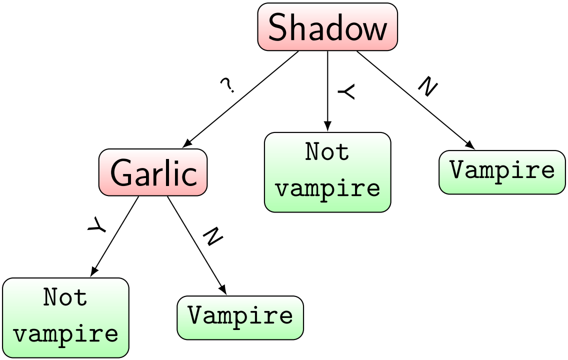


Garlic: 4 Complexion: 2 Accent: 0

Combined tests



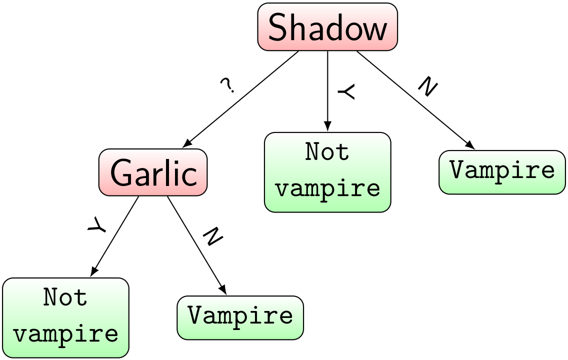
Combined tests



Problem

When our database size increases, none of the test likely to completely separate vampire from non-vampire. All tests will score 0 then.

Combined tests



Problem

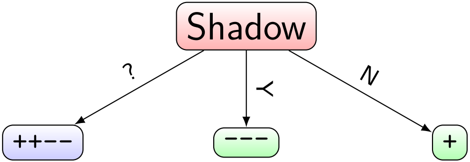
When our database size increases, none of the test likely to completely separate vampire from non-vampire. All tests will score 0 then.
Entropy comes to the rescue!

Conditional entropy as a measure of test efficiency

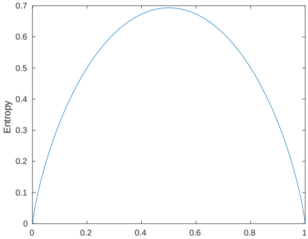
Consider the database is randomly sampled from a distribution. A set is

- Very homogeneous $\approx$ high certainty
- Not so homogenous $\approx$ high randomness

These can be measured with its entropy



$H(V|S=?)=1$ $H(V|S=Y)=0$ $H(V|S=N)=0$

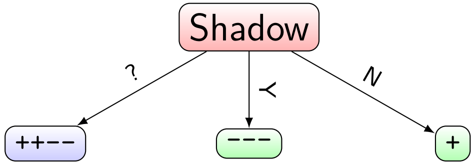


Conditional entropy as a measure of test efficiency

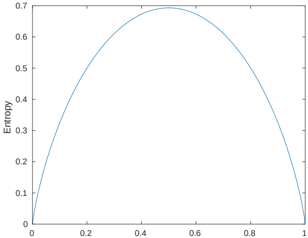
Consider the database is randomly sampled from a distribution. A set is

- Very homogeneous ≈ high certainty
- Not so homogenous ≈ high randomness

These can be measured with its entropy



$H(V|S=?)=1$ $\quad H(V|S=Y)=0$ $\quad H(V|S=N)=0$



Remaining uncertainty given the test:

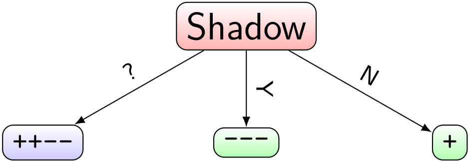
$$\frac{4}{8}H(V|S=?)$$

Conditional entropy as a measure of test efficiency

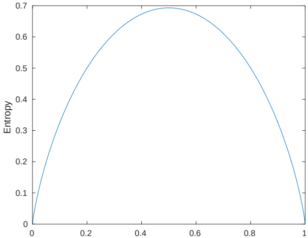
Consider the database is randomly sampled from a distribution. A set is

- Very homogeneous $\approx$ high certainty
- Not so homogenous $\approx$ high randomness

These can be measured with its entropy



$H(V|S=?)=1$ $H(V|S=Y)=0$ $H(V|S=N)=0$



Remaining uncertainty given the test:

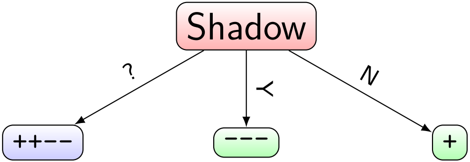
$$\frac{4}{8}H(V|S=?)+\frac{3}{8}H(V|S=Y)$$

Conditional entropy as a measure of test efficiency

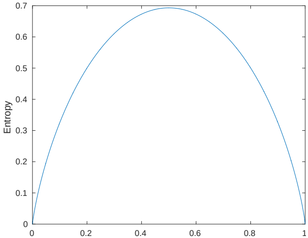
Consider the database is randomly sampled from a distribution. A set is

- Very homogeneous $\approx$ high certainty
- Not so homogenous $\approx$ high randomness

These can be measured with its entropy



$H(V|S=?)=1$ $H(V|S=Y)=0$ $H(V|S=N)=0$



Remaining uncertainty given the test:

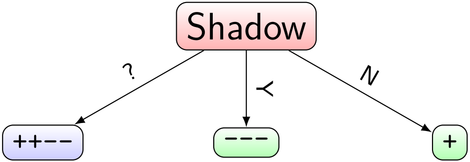
$$\frac{4}{8}H(V|S=?)+\frac{3}{8}H(V|S=Y)+\frac{1}{8}H(V|S=N)=0.5$$

Conditional entropy as a measure of test efficiency

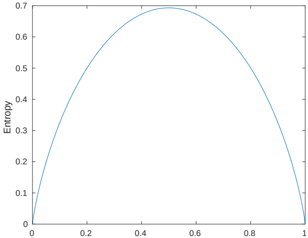
Consider the database is randomly sampled from a distribution. A set is

- Very homogeneous ≈ high certainty
- Not so homogenous ≈ high randomness

These can be measured with its entropy



$H(V|S=?)=1$ $H(V|S=Y)=0$ $H(V|S=N)=0$



Remaining uncertainty given the test:

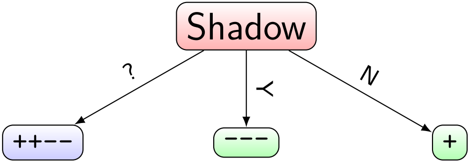
$$\frac{4}{8}H(V|S=?)+\frac{3}{8}H(V|S=Y)+\frac{1}{8}H(V|S=N)=0.5$$
$$=Pr(S=?)H(V|S=?)+Pr(S=Y)H(V|S=Y)+Pr(S=N)H(V|S=N)$$

Conditional entropy as a measure of test efficiency

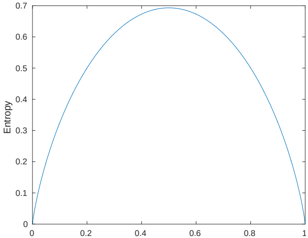
Consider the database is randomly sampled from a distribution. A set is

- Very homogeneous $\approx$ high certainty
- Not so homogenous $\approx$ high randomness

These can be measured with its entropy



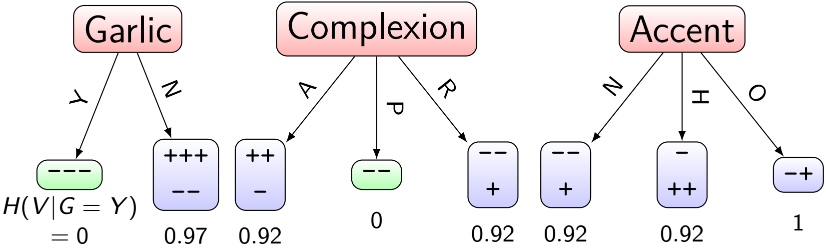
$H(V|S=?)=1$ $H(V|S=Y)=0$ $H(V|S=N)=0$



Remaining uncertainty given the test:

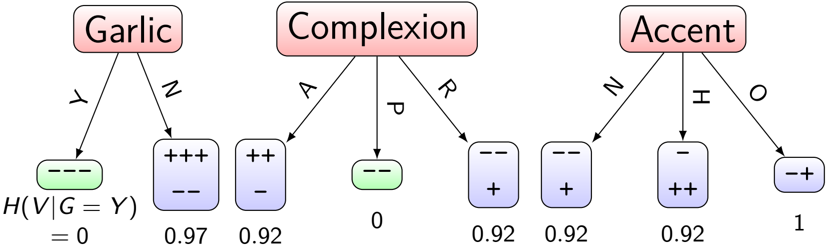
$$
\begin{aligned}
&\frac{4}{8}H(V|S=?)+\frac{3}{8}H(V|S=Y)+\frac{1}{8}H(V|S=N)=0.5\\
=&Pr(S=?)H(V|S=?)+Pr(S=Y)H(V|S=Y)+Pr(S=N)H(V|S=N)\\
=&H(V|S)
\end{aligned}
$$

Remaining uncertainty



$H(V|S) = 0.5$

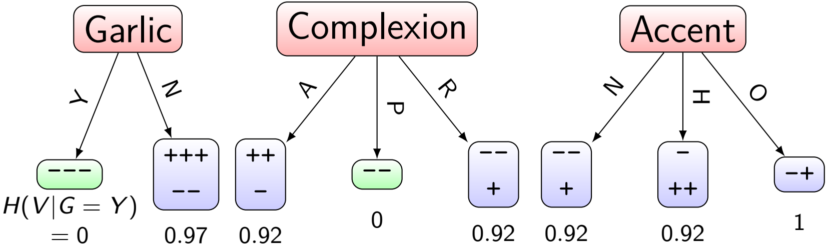
Remaining uncertainty



$$H(V|S) = 0.5$$

$$H(V|G) = \frac{3}{8}\cdot 0 + \frac{5}{8}\cdot 0.97 = 0.61$$

Remaining uncertainty

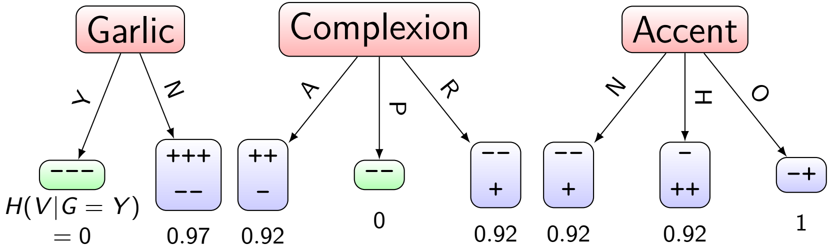


$$H(V|S) = 0.5$$

$$H(V|G) = \frac{3}{8}\cdot 0 + \frac{5}{8}\cdot 0.97 = 0.61$$

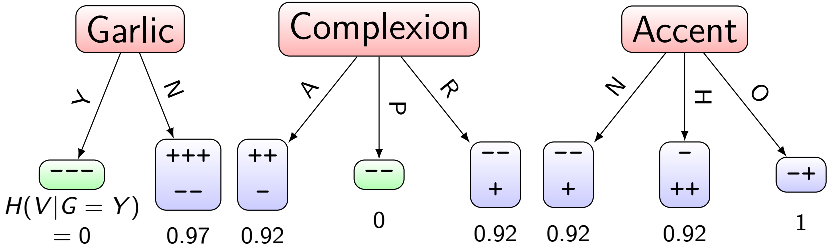
$$H(V|C) = \frac{3}{8}\cdot 0.92 + \frac{2}{8}\cdot 0 + \frac{3}{8}\cdot 0.92 = 0.69$$

Remaining uncertainty



$$
\begin{aligned}
H(V|S) =& 0.5 \\
H(V|G) =& \frac{3}{8} \cdot 0 + \frac{5}{8} \cdot 0.97 = 0.61 \\
H(V|C) =& \frac{3}{8} \cdot 0.92 + \frac{2}{8} \cdot 0 + \frac{3}{8} \cdot 0.92 = 0.69 \\
H(V|A) =& \frac{3}{8} \cdot 0.92 + \frac{3}{8} \cdot 0.92 + \frac{2}{8} \cdot 1 = 0.94
\end{aligned}
$$

Remaining uncertainty



$$
\begin{aligned}
H(V|S) =& 0.5\\
H(V|G) =& \frac{3}{8}\cdot 0 + \frac{5}{8}\cdot 0.97 = 0.61\\
H(V|C) =& \frac{3}{8}\cdot 0.92 + \frac{2}{8}\cdot 0 + \frac{3}{8}\cdot 0.92 = 0.69\\
H(V|A) =& \frac{3}{8}\cdot 0.92 + \frac{3}{8}\cdot 0.92 + \frac{2}{8}\cdot 1 = 0.94
\end{aligned}
$$

$H(V|S)$ is maximum. Thus should pick test S first

Potential extensions

- The test does not need to return discrete result. Let X be the test outcome. It can be continuous as well

Potential extensions

- The test does not need to return discrete result. Let X be the test outcome. It can be continuous as well
  - We should just pick i such that $H(V|X_i)$ to be as small as possible

Potential extensions

- The test does not need to return discrete result. Let X be the test outcome. It can be continuous as well
  - We should just pick i such that $H(V|X_i)$ to be as small as possible
  - It is equivalent of saying $I(V;X_i) = H(V) - H(V|X_i)$ is as large as possible. This is intuitive because we want to pick the information that is most relevant (sharing most information with) to V

Potential extensions

- The test does not need to return discrete result. Let X be the test outcome. It can be continuous as well
  - We should just pick i such that $H(V|X_i)$ to be as small as possible
  - It is equivalent of saying $I(V;X_i) = H(V) - H(V|X_i)$ is as large as possible. This is intuitive because we want to pick the information that is most relevant (sharing most information with) to V
- Build a number of trees instead of a single tree $\Rightarrow$ random forests

Random forests

- Pick random subset of training samples
- Train on each random subset but limited to a subset of features/attributes
- Given a test sample
    - Classify sample using each of the trees
    - Make final decision based on majority vote

Law of Large Number (LLN)

If we randomly sample $x_1, x_2, \cdots, x_N$ from an i.i.d. (identical and independently distributed) source, the average of $f(x_i)$ will approach the expected value as $N \to \infty$. That is,

$$\frac{1}{N}\sum_{i=1}^{N} f(x_i) = E[f(X)] \qquad \text{as } N \to \infty$$

Law of Large Number (LLN)

If we randomly sample $x_1, x_2, \cdots, x_N$ from an i.i.d. (identical and independently distributed) source, the average of $f(x_i)$ will approach the expected value as $N \to \infty$. That is,

$$\frac{1}{N}\sum_{i=1}^{N} f(x_i) = E[f(X)] \qquad \text{as } N \to \infty$$

Example

This is precisely how poll supposes to work! Pollster randomly draws sample from a portion of the population but will expect the prediction matches the outcome

Proof of LLN

The LLN is a rather strong result. We will only show a weak version here

$$Pr\left(\left|\frac{1}{N}\sum_{i=1}^{N} f(X_i) - E[f(X)]\right| \geq a\right) \leq \frac{Var(f(X))}{Na^2} \propto \frac{1}{N}$$

Proof of LLN

The LLN is a rather strong result. We will only show a weak version here

$$Pr\left(\left|\frac{1}{N}\sum_{i=1}^{N} f(X_i) - E[f(X)]\right| \geq a\right) \leq \frac{Var(f(X))}{Na^2} \propto \frac{1}{N}$$

Markov's Inequality

$$Pr(X \geq b) \leq \frac{E[X]}{b} \qquad \text{if } X \geq 0$$

Proof of LLN

The LLN is a rather strong result. We will only show a weak version here

$$Pr\left(\left|\frac{1}{N}\sum_{i=1}^{N} f(X_i) - E[f(X)]\right| \geq a\right) \leq \frac{Var(f(X))}{Na^2} \propto \frac{1}{N}$$

Markov's Inequality

$$Pr(X \geq b) \leq \frac{E[X]}{b} \qquad \text{if } X \geq 0$$

Proof:

$X = I(X \geq b) \cdot X + I(X < b) \cdot X \geq I(X \geq b) \cdot b$

Proof of LLN

The LLN is a rather strong result. We will only show a weak version here

$$Pr\left(\left|\frac{1}{N}\sum_{i=1}^{N} f(X_i) - E[f(X)]\right| \geq a\right) \leq \frac{Var(f(X))}{Na^2} \propto \frac{1}{N}$$

Markov's Inequality

$$Pr(X \geq b) \leq \frac{E[X]}{b} \qquad \text{if } X \geq 0$$

Proof:

$X = I(X \geq b) \cdot X + I(X < b) \cdot X \geq I(X \geq b) \cdot b \Rightarrow E[X] \geq Pr(X \geq b) \cdot b$

Proof of LLN

Markov's Inequality

$$Pr(X \geq b) \leq \frac{E[X]}{b} \qquad \text{if } X \geq 0$$

Chebyshev's Inequality

$$Pr(|Y - E[Y]| \geq a) \leq \frac{Var(Y)}{a^2}$$

Proof of LLN

Markov's Inequality

$$Pr(X \geq b) \leq \frac{E[X]}{b} \qquad \text{if } X \geq 0$$

Chebyshev's Inequality

$$Pr(|Y - E[Y]| \geq a) \leq \frac{Var(Y)}{a^2}$$

Proof: Take $X = |Y - E[Y]|^2$ and $b = a^2$, by Markov's Inequality

Proof of LLN

Markov's Inequality

$$Pr(X \geq b) \leq \frac{E[X]}{b} \qquad \text{if } X \geq 0$$

Chebyshev's Inequality

$$Pr(|Y - E[Y]| \geq a) \leq \frac{Var(Y)}{a^2}$$

Proof: Take $X = |Y - E[Y]|^2$ and $b = a^2$, by Markov's Inequality

$$Pr(|Y - E[Y]| \geq a) = Pr(|Y - E[Y]|^2 \geq a^2)$$

$$\leq \frac{E[|Y - E[Y]|^2]}{a^2}$$

Proof of LLN

Markov's Inequality

$$Pr(X \geq b) \leq \frac{E[X]}{b} \qquad \text{if } X \geq 0$$

Chebyshev's Inequality

$$Pr(|Y - E[Y]| \geq a) \leq \frac{Var(Y)}{a^2}$$

Proof: Take $X = |Y - E[Y]|^2$ and $b = a^2$, by Markov's Inequality

$$Pr(|Y - E[Y]| \geq a) = Pr(|Y - E[Y]|^2 \geq a^2)$$

$$\leq \frac{E[|Y - E[Y]|^2]}{a^2} = \frac{Var(Y)}{a^2}$$

Proof of LLN

Chebyshev's Inequality

$$Pr(|Y - E[Y]| \geq a) \leq \frac{Var(Y)}{a^2}$$

Proof of weak LLN

Let $Z_N = \frac{1}{N}\sum_{i=1}^{N} f(X_i)$, apparently $E[Z_N] = E[f(X)]$ and

$$Var(Z_N) = \frac{1}{N^2}\sum_{i=1}^{N} Var(f(X)) = \frac{Var(f(X))}{N}$$

By Chebyshev's Inequality,

Proof of LLN

Chebyshev's Inequality

$$Pr(|Y - E[Y]| \geq a) \leq \frac{Var(Y)}{a^2}$$

Proof of weak LLN

Let $Z_N = \frac{1}{N}\sum_{i=1}^N f(X_i)$, apparently $E[Z_N] = E[f(X)]$ and

$$Var(Z_N) = \frac{1}{N^2}\sum_{i=1}^N Var(f(X)) = \frac{Var(f(X))}{N}$$

By Chebyshev's Inequality,

$$Pr\left(\left|\frac{1}{N}\sum_{i=1}^N f(X_i) - E[f(X)]\right| \geq a\right)$$

$$=Pr(|Z_N - E[Z_N]| \geq a) \leq \frac{Var(Z_N)}{a^2}$$

Proof of LLN

Chebyshev's Inequality

$$Pr(|Y - E[Y]| \geq a) \leq \frac{Var(Y)}{a^2}$$

Proof of weak LLN

Let $Z_N = \frac{1}{N}\sum_{i=1}^N f(X_i)$, apparently $E[Z_N] = E[f(X)]$ and

$$Var(Z_N) = \frac{1}{N^2}\sum_{i=1}^N Var(f(X)) = \frac{Var(f(X))}{N}$$

By Chebyshev's Inequality,

$$Pr\left(\left|\frac{1}{N}\sum_{i=1}^N f(X_i) - E[f(X)]\right| \geq a\right)$$

$$= Pr(|Z_N - E[Z_N]| \geq a) \leq \frac{Var(Z_N)}{a^2} = \frac{Var(f(X))}{Na^2}$$

Main idea

Consider a sequence of symbols $x_1, x_2, \cdots, x_N$ sampled from a DMS and consider the sample average of the log-probabilities of each sampled symbols

$$\frac{1}{N}\sum_{i=1}^{N}\log\frac{1}{p(x_i)} \to E\left[\log\frac{1}{p(X)}\right]$$

by LLN.

Main idea

Consider a sequence of symbols $x_1, x_2, \cdots, x_N$ sampled from a DMS and consider the sample average of the log-probabilities of each sampled symbols

$$\frac{1}{N}\sum_{i=1}^{N} \log \frac{1}{p(x_i)} \to E\left[\log \frac{1}{p(X)}\right] = H(X)$$

by LLN.

Main idea

Consider a sequence of symbols $x_1, x_2, \cdots, x_N$ sampled from a DMS and consider the sample average of the log-probabilities of each sampled symbols

$$\frac{1}{N}\sum_{i=1}^{N}\log\frac{1}{p(x_i)} \to E\left[\log\frac{1}{p(X)}\right] = H(X)$$

by LLN. But for the LHS,

$$\frac{1}{N}\sum_{i=1}^{N}\log\frac{1}{p(x_i)} = \frac{1}{N}\log\frac{1}{\prod_{i=1}^{N}p(x_i)} = -\frac{1}{N}\log p(x^N),$$

where $x^N = x_1, x_2, \cdots, x_N$

Main idea

Consider a sequence of symbols $x_1, x_2, \cdots, x_N$ sampled from a DMS and consider the sample average of the log-probabilities of each sampled symbols

$$\frac{1}{N}\sum_{i=1}^{N}\log\frac{1}{p(x_i)} \to E\left[\log\frac{1}{p(X)}\right] = H(X)$$

by LLN. But for the LHS,

$$\frac{1}{N}\sum_{i=1}^{N}\log\frac{1}{p(x_i)} = \frac{1}{N}\log\frac{1}{\prod_{i=1}^{N}p(x_i)} = -\frac{1}{N}\log p(x^N),$$

where $x^N = x_1, x_2, \cdots, x_N$

Rearranging the terms, this implies that for any sequence sampled from the source, the probability of the sampled sequence $p(x^N) \to 2^{-NH(X)}$!

Set of typical sequences

Let's name the sequence x^N with $p(x^N) \sim 2^{-NH(X)}$ typical and define the set of typical sequences

$$\mathcal{A}_\epsilon^N(X) = \{x^N | 2^{-N(H(X)+\epsilon)} \leq p(x^N) \leq 2^{-N(H(X)-\epsilon)}\}$$

Set of typical sequences

Let's name the sequence x^N with $p(x^N) \sim 2^{-NH(X)}$ typical and define the set of typical sequences

$$\mathcal{A}_\epsilon^N(X) = \{x^N | 2^{-N(H(X)+\epsilon)} \leq p(x^N) \leq 2^{-N(H(X)-\epsilon)}\}$$

- For any $\epsilon > 0$, we can find a sufficiently large N such that any sampled sequence from the source is typical

Set of typical sequences

Let's name the sequence x^N with $p(x^N) \sim 2^{-NH(X)}$ typical and define the set of typical sequences

$$\mathcal{A}_\epsilon^N(X) = \{x^N | 2^{-N(H(X)+\epsilon)} \leq p(x^N) \leq 2^{-N(H(X)-\epsilon)}\}$$

- For any $\epsilon > 0$, we can find a sufficiently large N such that any sampled sequence from the source is typical
- Since all typical sequences have probability $\sim 2^{-NH(X)}$ and they fill up the entire probability space (everything is typical), there should be approximately $\frac{1}{2^{-NH(X)}} = 2^{NH(X)}$ typical sequences

Precise bounds on the size of typical set

$$(1-\delta)2^{N(H(X)-\epsilon)} \leq |\mathcal{A}_\epsilon^N(X)| \leq 2^{N(H(X)+\epsilon)}$$

$$1 \geq Pr(X^N \in \mathcal{A}_\epsilon^N(X))$$

Precise bounds on the size of typical set

$$(1-\delta)2^{N(H(X)-\epsilon)} \le |\mathcal{A}_\epsilon^N(X)| \le 2^{N(H(X)+\epsilon)}$$

$$1 \ge Pr(X^N \in \mathcal{A}_\epsilon^N(X)) = \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} p(x^N)$$

Precise bounds on the size of typical set

$$(1-\delta)2^{N(H(X)-\epsilon)} \leq |\mathcal{A}_\epsilon^N(X)| \leq 2^{N(H(X)+\epsilon)}$$

$$1 \geq Pr(X^N \in \mathcal{A}_\epsilon^N(X)) = \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} p(x^N) \geq \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} 2^{-N(H(X)+\epsilon)}$$

Precise bounds on the size of typical set

$$(1-\delta)2^{N(H(X)-\epsilon)} \leq |\mathcal{A}_\epsilon^N(X)| \leq 2^{N(H(X)+\epsilon)}$$

$$1 \geq Pr(X^N \in \mathcal{A}_\epsilon^N(X)) = \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} p(x^N) \geq \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} 2^{-N(H(X)+\epsilon)}$$

$$= |\mathcal{A}_\epsilon^N(X)| 2^{-N(H(X)+\epsilon)}$$

Precise bounds on the size of typical set

$$(1-\delta)2^{N(H(X)-\epsilon)} \leq |\mathcal{A}_\epsilon^N(X)| \leq 2^{N(H(X)+\epsilon)}$$

$$1 \geq Pr(X^N \in \mathcal{A}_\epsilon^N(X)) = \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} p(x^N) \geq \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} 2^{-N(H(X)+\epsilon)}$$
$$= |\mathcal{A}_\epsilon^N(X)| 2^{-N(H(X)+\epsilon)}$$

For a sufficiently large N, we have

$$1-\delta \leq Pr(X^N \in \mathcal{A}_\epsilon^N(X))$$

Precise bounds on the size of typical set

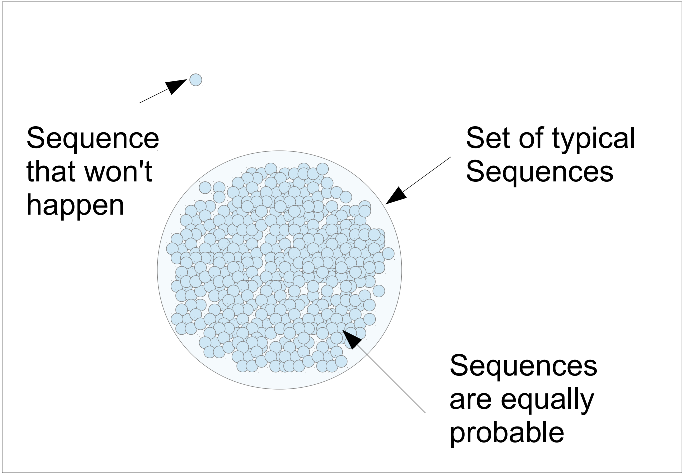
$$(1-\delta)2^{N(H(X)-\epsilon)} \leq |\mathcal{A}_\epsilon^N(X)| \leq 2^{N(H(X)+\epsilon)}$$

$$1 \geq Pr(X^N \in \mathcal{A}_\epsilon^N(X)) = \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} p(x^N) \geq \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} 2^{-N(H(X)+\epsilon)}$$
$$= |\mathcal{A}_\epsilon^N(X)| 2^{-N(H(X)+\epsilon)}$$

For a sufficiently large N, we have

$$1-\delta \leq Pr(X^N \in \mathcal{A}_\epsilon^N(X)) = \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} p(x^N) \leq \sum_{x^N \in \mathcal{A}_\epsilon^N(X)} 2^{-N(H(X)-\epsilon)}$$
$$= |\mathcal{A}_\epsilon^N(X)| 2^{-N(H(X)-\epsilon)}$$

AEP



Asymptotic equipatition refers to the fact that the probability space is equally partitioned by the typical sequences

AEP and compression limit

Consider coin flipping again, let say $Pr(Head) = 0.3$ and $N = 1000$

AEP and compression limit

Consider coin flipping again, let say $Pr(Head) = 0.3$ and $N = 1000$

- The typical sequences will be those with approximately 300 heads and 700 tails

AEP and compression limit

Consider coin flipping again, let say $Pr(Head) = 0.3$ and $N = 1000$

- The typical sequences will be those with approximately 300 heads and 700 tails
- AEP (LLN) tells us that it is almost impossible to get, say, a sequence of 100 heads and 900 tails

AEP and compression limit

Consider coin flipping again, let say $Pr(Head) = 0.3$ and $N = 1000$

- The typical sequences will be those with approximately 300 heads and 700 tails
- AEP (LLN) tells us that it is almost impossible to get, say, a sequence of 100 heads and 900 tails
- AEP also tells us that the number of typical sequences are approximately $2^{NH(X)}$

AEP and compression limit

Consider coin flipping again, let say $Pr(Head) = 0.3$ and $N = 1000$

- The typical sequences will be those with approximately 300 heads and 700 tails
- AEP (LLN) tells us that it is almost impossible to get, say, a sequence of 100 heads and 900 tails
- AEP also tells us that the number of typical sequences are approximately $2^{NH(X)}$
- Therefore, we can simply assign index to all the typical sequences and ignore the rest. Then we only need $\log 2^{NH(X)} = NH(X)$ to store a sequence of N symbols. And on average, we need $H(X)$ bits per symbol as before!

Previously...

- Identification/Decision trees
- Random forests
- Law of Large Number
- Asymptotic equipartition (AEP) and typical sequences

This time

- Joint typical sequences
- Covering and Packing Lemmas
- Channel coding setup
- Channel coding rate
- Channel capacity
- Channel Coding Theorem

Jointly typical sequences

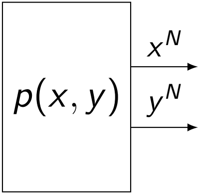
For a pair of sequences x^N and y^N, we say that they are jointly typical if

$$2^{-N(H(X,Y)+\epsilon)} \leq p(x^N, y^N) \leq 2^{-N(H(X,Y)-\epsilon)}$$

and x^N and y^N themselves are typical

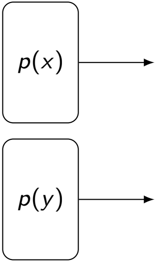
As in the single sequence case,

- Any sequence pair drawing from a joint source $p(x,y)$ is essentially jointly typical
- There are $\sim 2^{NH(X,Y)}$ jointly typical sequences



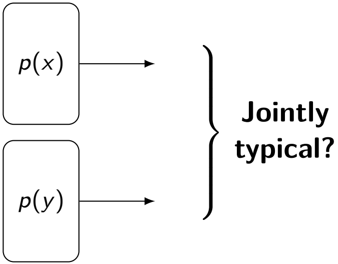
Joint typicality of independent seqences

- Given sequences X^N and Y^N independently drawn from discrete memoryless sources $p(x)$ and $p(y)$



Joint typicality of independent seqences

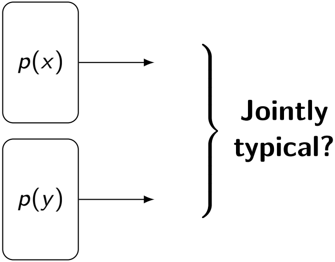
- Given sequences X^N and Y^N independently drawn from discrete memoryless sources $p(x)$ and $p(y)$
- What is the probability that X^N and Y^N are jointly typical?



Joint typicality of independent seqences

- Given sequences X^N and Y^N independently drawn from discrete memoryless sources $p(x)$ and $p(y)$
- What is the probability that X^N and Y^N are jointly typical?

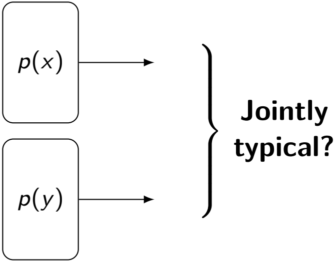
$$Pr((X^N, Y^N) \in \mathcal{A}_\epsilon^{(N)})$$
$$= \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} p(x^N, y^N)$$



Joint typicality of independent seqences

- Given sequences X^N and Y^N independently drawn from discrete memoryless sources $p(x)$ and $p(y)$
- What is the probability that X^N and Y^N are jointly typical?

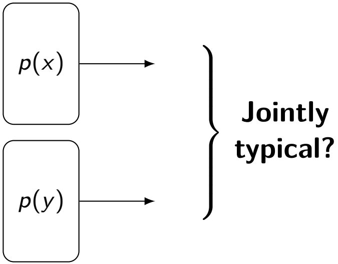
$$Pr((X^N, Y^N) \in \mathcal{A}_\epsilon^{(N)})$$
$$= \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} p(x^N, y^N)$$
$$= \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} p(x^N)p(y^N)$$



Joint typicality of independent seqences

- Given sequences X^N and Y^N independently drawn from discrete memoryless sources $p(x)$ and $p(y)$
- What is the probability that X^N and Y^N are jointly typical?

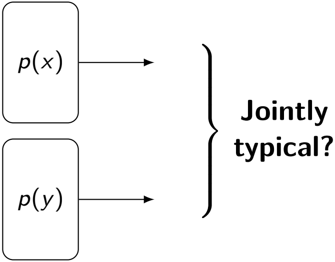
$$Pr((X^N, Y^N) \in \mathcal{A}_\epsilon^{(N)})$$
$$= \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} p(x^N, y^N)$$
$$= \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} p(x^N)p(y^N)$$
$$\leq \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} 2^{-N(H(X)-\epsilon)}2^{-N(H(Y)-\epsilon)}$$



Joint typicality of independent seqences

- Given sequences X^N and Y^N independently drawn from discrete memoryless sources $p(x)$ and $p(y)$
- What is the probability that X^N and Y^N are jointly typical?

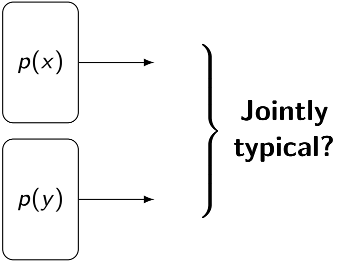
$$
\begin{aligned}
&Pr((X^N, Y^N) \in \mathcal{A}_\epsilon^{(N)}) \\
&= \sum_{\{(x^N, y^N) | (x^N, y^N) \in \mathcal{A}_\epsilon^{(N)}\}} p(x^N, y^N) \\
&= \sum_{\{(x^N, y^N) | (x^N, y^N) \in \mathcal{A}_\epsilon^{(N)}\}} p(x^N) p(y^N) \\
&\leq \sum_{\{(x^N, y^N) | (x^N, y^N) \in \mathcal{A}_\epsilon^{(N)}\}} 2^{-N(H(X)-\epsilon)} 2^{-N(H(Y)-\epsilon)} \\
&\leq 2^{-N(I(X;Y)-3\epsilon)}
\end{aligned}
$$



Joint typicality of independent seqences

- Given sequences X^N and Y^N independently drawn from discrete memoryless sources $p(x)$ and $p(y)$
- What is the probability that X^N and Y^N are jointly typical?

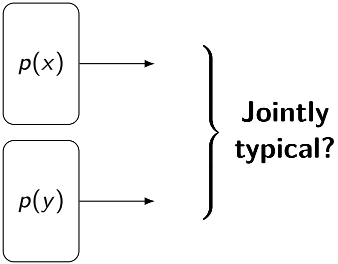
$$Pr((X^N, Y^N) \in \mathcal{A}_\epsilon^{(N)})$$
$$= \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} p(x^N, y^N)$$
$$= \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} p(x^N)p(y^N)$$



Joint typicality of independent seqences

- Given sequences X^N and Y^N independently drawn from discrete memoryless sources $p(x)$ and $p(y)$
- What is the probability that X^N and Y^N are jointly typical?

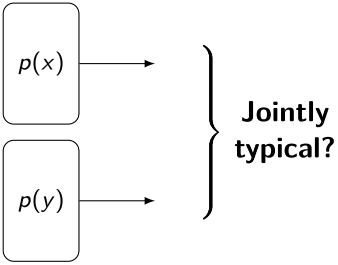
$$
\begin{aligned}
&Pr((X^N, Y^N) \in \mathcal{A}_\epsilon^{(N)}) \\
&= \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} p(x^N, y^N) \\
&= \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} p(x^N)p(y^N) \\
&\geq \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} 2^{-N(H(X)+\epsilon)}2^{-N(H(Y)+\epsilon)}
\end{aligned}
$$



Joint typicality of independent seqences

- Given sequences X^N and Y^N independently drawn from discrete memoryless sources $p(x)$ and $p(y)$
- What is the probability that X^N and Y^N are jointly typical?

$$
\begin{aligned}
&Pr((X^N,Y^N)\in\mathcal{A}_\epsilon^{(N)}) \\
&= \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} p(x^N,y^N) \\
&= \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} p(x^N)p(y^N) \\
&\geq \sum_{\{(x^N,y^N)|(x^N,y^N)\in\mathcal{A}_\epsilon^{(N)}\}} 2^{-N(H(X)+\epsilon)}2^{-N(H(Y)+\epsilon)} \\
&\geq (1-\delta)2^{-N(I(X;Y)+3\epsilon)}
\end{aligned}
$$



Packing lemma

How many independent Y^N sequences can pack with some X^N without becoming jointly typical with X^N?

- Say, M Y^N sequences were drawn

Packing lemma

How many independent Y^N sequences can pack with some X^N without becoming jointly typical with X^N?

- Say, M Y^N sequences were drawn
- The probability of any of Y^N to be jointly typical with X^N is bounded by

$$Pr(\text{Any one of } M \ Y^N \text{ jointly typical with } X^N)$$

Packing lemma

How many independent Y^N sequences can pack with some X^N without becoming jointly typical with X^N?

- Say, M Y^N sequences were drawn
- The probability of any of Y^N to be jointly typical with X^N is bounded by

$$
\begin{aligned}
&Pr(\text{Any one of } M \ Y^N \text{ jointly typical with } X^N) \\
\leq & M Pr((X^N, Y^N) \in \mathcal{A}_\epsilon^N(X, Y)) \\
\leq & M 2^{-N(I(X;Y)-3\epsilon)}
\end{aligned}
$$

Packing lemma

How many independent Y^N sequences can pack with some X^N without becoming jointly typical with X^N?

- Say, M Y^N sequences were drawn
- The probability of any of Y^N to be jointly typical with X^N is bounded by

$$
\begin{aligned}
&Pr(\text{Any one of } M\ Y^N \text{ jointly typical with } X^N)\\
\leq & MPr((X^N, Y^N) \in \mathcal{A}_\epsilon^N(X, Y))\\
\leq & M2^{-N(I(X;Y)-3\epsilon)}\\
\leq & 2^{-N(I(X;Y)-R-3\epsilon)}
\end{aligned}
$$

where $2^{NR} = M$

Packing lemma

How many independent Y^N sequences can pack with some X^N without becoming jointly typical with X^N?

- Say, M Y^N sequences were drawn
- The probability of any of Y^N to be jointly typical with X^N is bounded by

$$
\begin{aligned}
&Pr(\text{Any one of } M\ Y^N \text{ jointly typical with } X^N)\\
\leq &MPr((X^N, Y^N) \in \mathcal{A}_\epsilon^N(X, Y))\\
\leq &M2^{-N(I(X;Y)-3\epsilon)}\\
\leq &2^{-N(I(X;Y)-R-3\epsilon)} \rightarrow 0 \text{ as } N \rightarrow \infty \text{ and } I(X;Y) - 3\epsilon > R,
\end{aligned}
$$

where $2^{NR} = M$

Packing lemma

How many independent Y^N sequences can pack with some X^N without becoming jointly typical with X^N?

- Say, M Y^N sequences were drawn
- The probability of any of Y^N to be jointly typical with X^N is bounded by

$$\begin{aligned}
&Pr(\text{Any one of } M\ Y^N \text{ jointly typical with } X^N)\\
\leq & MPr((X^N, Y^N) \in \mathcal{A}_\epsilon^N(X,Y))\\
\leq & M2^{-N(I(X;Y)-3\epsilon)}\\
\leq & 2^{-N(I(X;Y)-R-3\epsilon)} \to 0 \text{ as } N \to \infty \text{ and } I(X;Y) - 3\epsilon > R,
\end{aligned}$$

where $2^{NR} = M$

Since ϵ can be made arbitrarily small as N increases, as long as $I(X;Y) > R$, we can find a sufficiently large N so that we can "pack" the M Y^N with X^N and none of the Y^N will be jointly typical with X^N

Covering lemma

How many independent Y^N are needed until it is jointly typical with X^N?

- Again, draw $M(=2^{NR})$ Y^N sequences
- Under what condition that *at least one* Y^N jointly typical with X^N?

Covering lemma

How many independent Y^N are needed until it is jointly typical with X^N?

- Again, draw $M(= 2^{NR})$ Y^N sequences
- Under what condition that *at least one* Y^N jointly typical with X^N?

$$Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(X, Y) \text{ for all } m)$$

Covering lemma

How many independent Y^N are needed until it is jointly typical with X^N?

- Again, draw $M(= 2^{NR})$ Y^N sequences
- Under what condition that *at least one* Y^N jointly typical with X^N?

$$Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(X, Y) \text{ for all } m)$$

$$= \prod_{m=1}^{M} Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(Y, X))$$

Covering lemma

How many independent Y^N are needed until it is jointly typical with X^N?

- Again, draw $M(=2^{NR})$ Y^N sequences
- Under what condition that *at least one* Y^N jointly typical with X^N?

$$Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(X,Y) \text{ for all } m)$$

$$= \prod_{m=1}^{M} Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(Y,X))$$

$$= \prod_{m=1}^{M} \left[1 - Pr((X^N(m), Y^N) \in \mathcal{A}_\epsilon^{(N)}(Y,X))\right]$$

Covering lemma

How many independent Y^N are needed until it is jointly typical with X^N?

- Again, draw $M(=2^{NR})$ Y^N sequences
- Under what condition that *at least one* Y^N jointly typical with X^N?

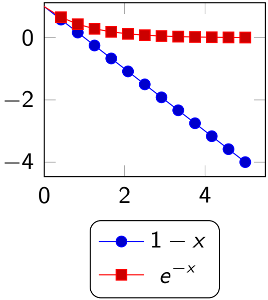
$$
\begin{aligned}
& Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(X, Y) \text{ for all } m) \\
=& \prod_{m=1}^{M} Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(Y, X)) \\
=& \prod_{m=1}^{M} \left[1 - Pr((X^N(m), Y^N) \in \mathcal{A}_\epsilon^{(N)}(Y, X))\right] \\
\leq& (1-(1-\delta)2^{-N(I(Y;X)+3\epsilon)})^M
\end{aligned}
$$

Covering lemma

How many independent Y^N are needed until it is jointly typical with X^N?

- Again, draw $M(=2^{NR})$ Y^N sequences
- Under what condition that *at least one* Y^N jointly typical with X^N?

$$
\begin{aligned}
& Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(X,Y) \text{ for all } m) \\
=& \prod_{m=1}^{M} Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(Y,X)) \\
=& \prod_{m=1}^{M} \left[1 - Pr((X^N(m), Y^N) \in \mathcal{A}_\epsilon^{(N)}(Y,X))\right] \\
\leq& (1-(1-\delta)2^{-N(I(Y;X)+3\epsilon)})^M \\
\leq& \exp(-M(1-\delta)2^{-N(I(Y;X)+3\epsilon)})
\end{aligned}
$$

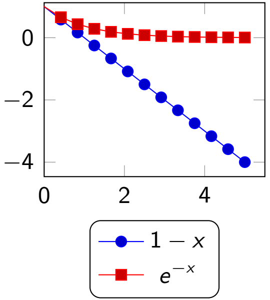


Covering lemma

How many independent Y^N are needed until it is jointly typical with X^N?

- Again, draw $M(=2^{NR})$ Y^N sequences
- Under what condition that *at least one* Y^N jointly typical with X^N?

$$
\begin{aligned}
& Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(X,Y) \text{ for all } m) \\
&= \prod_{m=1}^{M} Pr((X^N(m), Y^N) \notin \mathcal{A}_\epsilon^{(N)}(Y,X)) \\
&= \prod_{m=1}^{M} \left[1 - Pr((X^N(m), Y^N) \in \mathcal{A}_\epsilon^{(N)}(Y,X))\right] \\
&\leq (1-(1-\delta)2^{-N(I(Y;X)+3\epsilon)})^M \\
&\leq \exp(-M(1-\delta)2^{-N(I(Y;X)+3\epsilon)}) \\
&\leq \exp(-(1-\delta)2^{-N(I(Y;X)-R+3\epsilon)}) \to 0 \text{ as } N \to \infty \text{ and } R > I(X;Y)+3\epsilon
\end{aligned}
$$



Summary of packing lemma and covering lemma

Packing Lemma

We can "pack" $M = 2^{NR}$ (with $R < I(X;Y)$) x^N together without being jointly typical with y^N

Covering Lemma

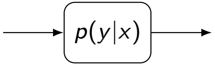
We can "cover" with $M = 2^{NR}$ (with $R > I(X;Y)$) x^N such that at least one x^N being jointly typical with y^N

Remark

- Packing lemma is useful in the proof of channel coding theorem
- Covering lemma is useful in the proof of rate-distortion theorem

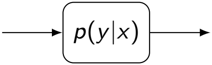
We will look into the above applications later in this course

Channel coding setup



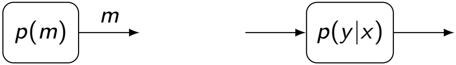
- As the name suggests, the output of a discrete memoryless channel (DMS) only depends on the current input (thus no memoryless). And both its input X and output Y are characterized by the conditional probability $p(y|x)$

Channel coding setup



- As the name suggests, the output of a discrete memoryless channel (DMS) only depends on the current input (thus no memoryless). And both its input X and output Y are characterized by the conditional probability $p(y|x)$
- Given an input sequence $x^N = x_1, \cdots, x_N$, the probability of getting an output sequence $y^N = y_1, \cdots, y_N$ is $p(y^N|x^N) = \prod_{i=1}^N p(y_i|x_i)$

Channel coding setup



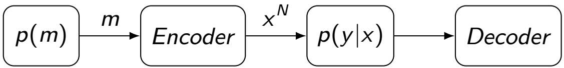
- As the name suggests, the output of a discrete memoryless channel (DMS) only depends on the current input (thus no memoryless). And both its input X and output Y are characterized by the conditional probability $p(y|x)$
- Given an input sequence $x^N = x_1, \cdots, x_N$, the probability of getting an output sequence $y^N = y_1, \cdots, y_N$ is $p(y^N|x^N) = \prod_{i=1}^N p(y_i|x_i)$
- Given a message m (say generated from a distribution $p(m)$)

Channel coding setup

$$p(m) \xrightarrow{m} \textit{Encoder} \longrightarrow p(y|x) \longrightarrow \textit{Decoder}$$

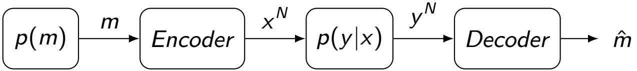
- As the name suggests, the output of a discrete memoryless channel (DMS) only depends on the current input (thus no memoryless). And both its input X and output Y are characterized by the conditional probability $p(y|x)$
- Given an input sequence $x^N = x_1, \cdots, x_N$, the probability of getting an output sequence $y^N = y_1, \cdots, y_N$ is $p(y^N|x^N) = \prod_{i=1}^N p(y_i|x_i)$
- Given a message m (say generated from a distribution $p(m)$)
    - We will have an encoder decoder pair

Channel coding setup



- As the name suggests, the output of a discrete memoryless channel (DMS) only depends on the current input (thus no memoryless). And both its input X and output Y are characterized by the conditional probability $p(y|x)$
- Given an input sequence $x^N = x_1, \cdots, x_N$, the probability of getting an output sequence $y^N = y_1, \cdots, y_N$ is $p(y^N|x^N) = \prod_{i=1}^N p(y_i|x_i)$
- Given a message m (say generated from a distribution $p(m)$)
  - We will have an encoder decoder pair
  - The encoder will convert m to x^N suitable for transmission

Channel coding setup



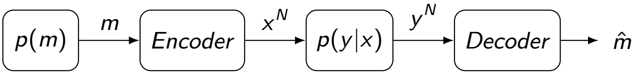
- As the name suggests, the output of a discrete memoryless channel (DMS) only depends on the current input (thus no memoryless). And both its input X and output Y are characterized by the conditional probability $p(y|x)$
- Given an input sequence $x^N = x_1, \cdots, x_N$, the probability of getting an output sequence $y^N = y_1, \cdots, y_N$ is $p(y^N|x^N) = \prod_{i=1}^N p(y_i|x_i)$
- Given a message m (say generated from a distribution $p(m)$)
    - We will have an encoder decoder pair
    - The encoder will convert m to x^N suitable for transmission
    - Decoder will try to extracted the message from the channel output y^N

Channel coding rate

$$\boxed{p(m)} \xrightarrow{m} \boxed{Encoder} \xrightarrow{x^N} \boxed{p(y|x)} \xrightarrow{y^N} \boxed{Decoder} \longrightarrow \hat{m}$$

The channel coding rate is defined as number of bits of message can be sent per channel use

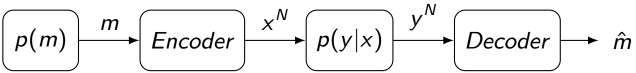
Channel coding rate



The channel coding rate is defined as number of bits of message can be sent per channel use

- Since there is $H(M)$ bits of information for each message M sent

Channel coding rate



The channel coding rate is defined as number of bits of message can be sent per channel use

- Since there is $H(M)$ bits of information for each message M sent
- $R = \frac{H(M)}{N}$

Channel capacity

- By Shannon's channel coding theorem, the capacity of the channel (will be shown later) is given by

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

Channel capacity

- By Shannon's channel coding theorem, the capacity of the channel (will be shown later) is given by

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

- This means that as long as the rate R is less than the capacity C, we can find encoder-decoder pair such that the decoding error $(Pr(\hat{M} \neq M))$ can be made arbitrarily small

Channel capacity

- By Shannon's channel coding theorem, the capacity of the channel (will be shown later) is given by

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

- This means that as long as the rate R is less than the capacity C, we can find encoder-decoder pair such that the decoding error $(Pr(\hat{M} \neq M))$ can be made arbitrarily small
- On the other hand, if R is larger than the capacity C, no matter how we try, it is impossible to recontruct m error free

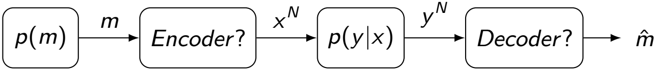
Channel capacity

- By Shannon's channel coding theorem, the capacity of the channel (will be shown later) is given by

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

- This means that as long as the rate R is less than the capacity C, we can find encoder-decoder pair such that the decoding error ($Pr(\hat{M} \neq M)$) can be made arbitrarily small
- On the other hand, if R is larger than the capacity C, no matter how we try, it is impossible to recontruct m error free
- An intuitive interpretation is that the amount of information can be passed through a channel is just mutual information between the input and output. And since we can pick the statistics of our input, we may make our choice wisely and maximize the mutual information. And the maximum that we can attain is the capacity

Continuous channel



Continuous channel

$$p(m) \xrightarrow{m} \textit{Encoder} \xrightarrow{x_\Delta^N} D/A \xrightarrow{x^N} p(y|x) \xrightarrow{y^N} A/D \xrightarrow{y_\Delta^N} \textit{Decoder} \longrightarrow \hat{m}$$

- For continuous channel, we can create a "pseudo" discrete channel using A/D and D/A converters

Continuous channel

$$p(m) \xrightarrow{m} \textit{Encoder} \xrightarrow{x_\Delta^N} \textcolor{red}{D/A} \xrightarrow{x^N} p(y|x) \xrightarrow{y^N} \textcolor{red}{A/D} \xrightarrow{y_\Delta^N} \textit{Decoder} \longrightarrow \hat{m}$$

- For continuous channel, we can create a "pseudo" discrete channel using A/D and D/A converters
- The maximum information that can pass through the channel will then be

$$\begin{aligned} C_\Delta &= \max_{p(x)} I(X_\Delta; Y_\Delta) = \max_{p(x)} H(Y_\Delta) - H(Y_\Delta|X_\Delta) \\ &\approx \max_{p(x)} h(Y) - \log\Delta - h(Y|X_\Delta) + \log\Delta \\ &\approx \max_{p(x)} h(Y) - h(Y|X) = \max_{p(x)} I(X;Y) \end{aligned}$$

Continuous channel

$$p(m) \xrightarrow{m} \text{Encoder} \xrightarrow{x_\Delta^N} D/A \xrightarrow{x^N} p(y|x) \xrightarrow{y^N} A/D \xrightarrow{y_\Delta^N} \text{Decoder} \longrightarrow \hat{m}$$

- For continuous channel, we can create a “pseudo” discrete channel using A/D and D/A converters
- The maximum information that can pass through the channel will then be

$$\begin{aligned} C_\Delta &= \max_{p(x)} I(X_\Delta; Y_\Delta) = \max_{p(x)} H(Y_\Delta) - H(Y_\Delta|X_\Delta) \\ &\approx \max_{p(x)} h(Y) - \log\Delta - h(Y|X_\Delta) + \log\Delta \\ &\approx \max_{p(x)} h(Y) - h(Y|X) = \max_{p(x)} I(X;Y) \end{aligned}$$

- As $\Delta \to 0$, $C = \max_{p(x)} I(X;Y)$. So expression is completely the same as the discrete case

Example: Binary symmetric channel

- Both input and output are binary (say take value 0 or 1)

Example: Binary symmetric channel

- Both input and output are binary (say take value 0 or 1)
- The channel is symmetric in the sense that

$$p_{Y|X}(1|0) = p_{Y|X}(0|1) = p$$

and

$$p_{Y|X}(0|0) = p_{Y|X}(1|1) = 1 - p,$$

where p is known to be the cross-over probability

Example: Binary symmetric channel

- Both input and output are binary (say take value 0 or 1)
- The channel is symmetric in the sense that

$$p_{Y|X}(1|0) = p_{Y|X}(0|1) = p$$

  and

$$p_{Y|X}(0|0) = p_{Y|X}(1|1) = 1 - p,$$

  where p is known to be the cross-over probability
- Capacity is given by

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

Example: Binary symmetric channel

- Both input and output are binary (say take value 0 or 1)
- The channel is symmetric in the sense that

$$p_{Y|X}(1|0) = p_{Y|X}(0|1) = p$$

and

$$p_{Y|X}(0|0) = p_{Y|X}(1|1) = 1 - p,$$

where p is known to be the cross-over probability

- Capacity is given by

$$\begin{aligned} C &= \max_{p(x)} I(X;Y) \\ &= \max_{p(x)} H(Y) - H(Y|X) \end{aligned}$$

Example: Binary symmetric channel

- Both input and output are binary (say take value 0 or 1)
- The channel is symmetric in the sense that

$$p_{Y|X}(1|0) = p_{Y|X}(0|1) = p$$

and

$$p_{Y|X}(0|0) = p_{Y|X}(1|1) = 1 - p,$$

where p is known to be the cross-over probability

- Capacity is given by

$$\begin{aligned} C &= \max_{p(x)} I(X;Y) \\ &= \max_{p(x)} H(Y) - H(Y|X) \\ &= \max_{p(x)} H(Y) - H(p) \end{aligned}$$

Example: Binary symmetric channel

- Both input and output are binary (say take value 0 or 1)
- The channel is symmetric in the sense that

$$p_{Y|X}(1|0) = p_{Y|X}(0|1) = p$$

and

$$p_{Y|X}(0|0) = p_{Y|X}(1|1) = 1 - p,$$

where p is known to be the cross-over probability

- Capacity is given by

$$\begin{aligned} C &= \max_{p(x)} I(X;Y) \\ &= \max_{p(x)} H(Y) - H(Y|X) \\ &= \max_{p(x)} H(Y) - H(p) = 1 - H(p) \end{aligned}$$

Example: Gaussian channel

The channel output $Y = X + Z$, where Z is a zero-mean Gaussian noise (independent of the input X)

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

Example: Gaussian channel

The channel output $Y = X + Z$, where Z is a zero-mean Gaussian noise (independent of the input X)

$$
\begin{aligned}
C &= \max_{p(x)} I(X;Y) \\
&= \max_{p(x)} h(Y) - h(Y|X)
\end{aligned}
$$

Example: Gaussian channel

The channel output $Y = X + Z$, where Z is a zero-mean Gaussian noise (independent of the input X)

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

Example: Gaussian channel

The channel output $Y = X + Z$, where Z is a zero-mean Gaussian noise (independent of the input X)

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

Example: Gaussian channel

The channel output $Y = X + Z$, where Z is a zero-mean Gaussian noise (independent of the input X)

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

Example: Gaussian channel

The channel output $Y = X + Z$, where Z is a zero-mean Gaussian noise (independent of the input X)

$$
\begin{aligned}
C &= \max_{p(x)} I(X;Y) \\
&= \max_{p(x)} h(Y) - h(Y|X) = \max_{p(x)} h(Y) - h(X+Z|X) \\
&= \max_{p(x)} h(Y) - h(Z|X) = \max_{p(x)} h(Y) - h(Z) \\
&= \max_{p(x)} h(Y) - \frac{1}{2}\log 2\pi e \sigma_Z^2
\end{aligned}
$$

Example: Gaussian channel

The channel output $Y = X + Z$, where Z is a zero-mean Gaussian noise (independent of the input X)

$$
\begin{aligned}
C &= \max_{p(x)} I(X;Y) \\
&= \max_{p(x)} h(Y) - h(Y|X) = \max_{p(x)} h(Y) - h(X+Z|X) \\
&= \max_{p(x)} h(Y) - h(Z|X) = \max_{p(x)} h(Y) - h(Z) \\
&= \max_{p(x)} h(Y) - \frac{1}{2}\log 2\pi e\sigma_Z^2 = \frac{1}{2}\log 2\pi e\sigma_Y^2 - \frac{1}{2}\log 2\pi e\sigma_Z^2
\end{aligned}
$$

Example: Gaussian channel

The channel output $Y = X + Z$, where Z is a zero-mean Gaussian noise (independent of the input X)

$$\begin{aligned}
C &= \max_{p(x)} I(X;Y) \\
&= \max_{p(x)} h(Y) - h(Y|X) = \max_{p(x)} h(Y) - h(X+Z|X) \\
&= \max_{p(x)} h(Y) - h(Z|X) = \max_{p(x)} h(Y) - h(Z) \\
&= \max_{p(x)} h(Y) - \frac{1}{2}\log 2\pi e\sigma_Z^2 = \frac{1}{2}\log 2\pi e\sigma_Y^2 - \frac{1}{2}\log 2\pi e\sigma_Z^2 \\
&= \frac{1}{2}\log\frac{\sigma_X^2+\sigma_Z^2}{\sigma_Z^2} = \frac{1}{2}\log\left(1+\frac{\sigma_X^2}{\sigma_Z^2}\right)
\end{aligned}$$

Example: Gaussian channel

The channel output $Y = X + Z$, where Z is a zero-mean Gaussian noise (independent of the input X)

$$
\begin{aligned}
C &= \max_{p(x)} I(X;Y) \\
&= \max_{p(x)} h(Y) - h(Y|X) = \max_{p(x)} h(Y) - h(X+Z|X) \\
&= \max_{p(x)} h(Y) - h(Z|X) = \max_{p(x)} h(Y) - h(Z) \\
&= \max_{p(x)} h(Y) - \frac{1}{2}\log 2\pi e\sigma_Z^2 = \frac{1}{2}\log 2\pi e\sigma_Y^2 - \frac{1}{2}\log 2\pi e\sigma_Z^2 \\
&= \frac{1}{2}\log\frac{\sigma_X^2+\sigma_Z^2}{\sigma_Z^2} = \frac{1}{2}\log\left(1+\frac{\sigma_X^2}{\sigma_Z^2}\right) = \frac{1}{2}\log(1+SNR),
\end{aligned}
$$

where SNR is the signal to noise ratio

Example: Additive White Gaussian Noise (AWGN) channel

Consider an AWGN channel with bandwidth W and two-sided power spectrum density of $N_0/2$

Example: Additive White Gaussian Noise (AWGN) channel

Consider an AWGN channel with bandwidth W and two-sided power spectrum density of $N_0/2$

- From the result of Nyquist and Shannon, a signal of bandwidth W will need to at least $2W$ samples per second to be fully reconstructed

Example: Additive White Gaussian Noise (AWGN) channel

Consider an AWGN channel with bandwidth W and two-sided power spectrum density of $N_0/2$

- From the result of Nyquist and Shannon, a signal of bandwidth W will need to at least $2W$ samples per second to be fully reconstructed
- Per each second, $2W$ samples needed to recover the signal

Example: Additive White Gaussian Noise (AWGN) channel

Consider an AWGN channel with bandwidth W and two-sided power spectrum density of $N_0/2$

- From the result of Nyquist and Shannon, a signal of bandwidth W will need to at least $2W$ samples per second to be fully reconstructed
- Per each second, $2W$ samples needed to recover the signal
- Per each second, $2W$ degrees of freedom exists $\Rightarrow 2W$ parallel Gaussian channel per second

Example: Additive White Gaussian Noise (AWGN) channel

Consider an AWGN channel with bandwidth W and two-sided power spectrum density of $N_0/2$

- From the result of Nyquist and Shannon, a signal of bandwidth W will need to at least $2W$ samples per second to be fully reconstructed
- Per each second, $2W$ samples needed to recover the signal
- Per each second, $2W$ degrees of freedom exists $\Rightarrow 2W$ parallel Gaussian channel per second
- Given N_0, $SNR = \frac{\sigma_X^2}{WN_0} = \frac{P}{WN_0}$

Example: Additive White Gaussian Noise (AWGN) channel

Consider an AWGN channel with bandwidth W and two-sided power spectrum density of $N_0/2$

- From the result of Nyquist and Shannon, a signal of bandwidth W will need to at least $2W$ samples per second to be fully reconstructed
- Per each second, $2W$ samples needed to recover the signal
- Per each second, $2W$ degrees of freedom exists $\Rightarrow 2W$ parallel Gaussian channel per second
- Given N_0, $SNR = \frac{\sigma_X^2}{WN_0} = \frac{P}{WN_0}$

$$C = 2W\frac{1}{2}\log(1+SNR) = W\log\left(1+\frac{P}{WN_0}\right)$$

Codebook construction

Forward statement

If the code rate $R < C = \max_{p(x)} I(X;Y)$, according to the Channel Coding Theorem, we should be able to find a code with encoding mapping $\mathbf{c} : m \in \{1, 2, \cdots, 2^{NR}\} \to \{0, 1\}^N$ and the error probability of transmitting any message $m \in \{1, 2, \cdots, 2^{NR}\}$, $p_e(m)$, is arbitrarily small

Codebook construction

Forward statement

If the code rate $R < C = \max_{p(x)} I(X;Y)$, according to the Channel Coding Theorem, we should be able to find a code with encoding mapping $\mathbf{c} : m \in \{1, 2, \cdots, 2^{NR}\} \rightarrow \{0, 1\}^N$ and the error probability of transmitting any message $m \in \{1, 2, \cdots, 2^{NR}\}$, $p_e(m)$, is arbitrarily small

- The main tool of the proof is random coding

Codebook construction

Forward statement

If the code rate $R < C = \max_{p(x)} I(X;Y)$, according to the Channel Coding Theorem, we should be able to find a code with encoding mapping $\mathbf{c}: m \in \{1,2,\cdots,2^{NR}\} \to \{0,1\}^N$ and the error probability of transmitting any message $m \in \{1,2,\cdots,2^{NR}\}$, $p_e(m)$, is arbitrarily small

- The main tool of the proof is random coding
- Let $p^*(x) = \arg\max_{p(x)} I(X;Y)$. Generate codewords from the DMS $p^*(x)$ by sampling 2^n length-n sequences from the source:

$$
\begin{aligned}
\mathbf{c}(1) =& (x_1(1), x_2(1), \cdots, x_N(1)) \\
\mathbf{c}(2) =& (x_1(2), x_2(2), \cdots, x_N(2)) \\
& \cdots \\
\mathbf{c}(2^{NR}) =& (x_1(2^{NR}), x_2(2^{NR}), \cdots, x_N(2^{NR}))
\end{aligned}
$$

Encoding and decoding

The encoding and decoding procedures will be as follows.

Encoding and decoding

The encoding and decoding procedures will be as follows.

Encoding

For input message m, output $\mathbf{c}(m) = (x_1(m), x_2(m), \cdots, x_N(m))$

Encoding and decoding

The encoding and decoding procedures will be as follows.

Encoding

For input message m, output $\mathbf{c}(m) = (x_1(m), x_2(m), \cdots, x_N(m))$

Decoding

Upon receiving sequence $\mathbf{y} = (y_1, y_2, \cdots, y_N)$, pick the sequence $\mathbf{c}(m)$ from $\{\mathbf{c}(1), \cdots, \mathbf{c}(2^{NR})\}$ such that $(\mathbf{c}(m), \mathbf{y})$ are jointly typical. That is $p_{X^N,Y^N}(\mathbf{c}(m), \mathbf{y}) \sim 2^{-nH(X,Y)}$. If no such $\mathbf{c}(m)$ exists or more than one such sequence exist, announce error. Otherwise output the decoded message as m

Average performance

Let us assume $M = m$, decoding error occurs when:

Average performance

Let us assume $M = m$, decoding error occurs when:

1. $P_1 = Pr(\mathbf{C}(m), \mathbf{Y}) \notin A_\epsilon^N(X, Y))$

Average performance

Let us assume $M = m$, decoding error occurs when:

1. $P_1 = Pr(\mathbf{C}(m), \mathbf{Y}) \notin A_\epsilon^N(X, Y))$
2. $P_2 : \exists M' \neq m$ and $(\mathbf{c}(M'), \mathbf{Y}) \in A_\epsilon^N(X, Y)$

Thus $p(error) = P(error|M = m) \leq P_1 + P_2$

Average performance

Let us assume $M = m$, decoding error occurs when:

1. $P_1 = Pr(\mathbf{C}(m), \mathbf{Y}) \notin A_\epsilon^N(X, Y))$
2. $P_2 : \exists M' \neq m$ and $(\mathbf{c}(M'), \mathbf{Y}) \in A_\epsilon^N(X, Y)$

Thus $p(error) = P(error|M = m) \leq P_1 + P_2$

1. Since $(\mathbf{C}(m), \mathbf{Y})$ is coming out of the joint source X, Y, $P_1 \to 0$ as $n \to \infty$

Average performance

Let us assume $M = m$, decoding error occurs when:

1. $P_1 = Pr(\mathbf{C}(m), \mathbf{Y}) \notin A_\epsilon^N(X, Y))$
2. $P_2 : \exists M' \neq m$ and $(\mathbf{c}(M'), \mathbf{Y}) \in A_\epsilon^N(X, Y)$

Thus $p(error) = P(error|M = m) \leq P_1 + P_2$

1. Since $(\mathbf{C}(m), \mathbf{Y})$ is coming out of the joint source X, Y, $P_1 \to 0$ as $n \to \infty$
2. Note that $\mathbf{C}(M')$ and $\mathbf{Y}$ are independent and thus by Packing lemma,

$$P_2 \leq 2^{-N(I(X;Y)-R-3\epsilon)} \tag{1}$$

Average performance

Let us assume $M = m$, decoding error occurs when:

1. $P_1 = Pr(\mathbf{C}(m), \mathbf{Y}) \notin A_\epsilon^N(X, Y))$
2. $P_2 : \exists M' \neq m$ and $(\mathbf{c}(M'), \mathbf{Y}) \in A_\epsilon^N(X, Y)$

Thus $p(error) = P(error|M = m) \leq P_1 + P_2$

1. Since $(\mathbf{C}(m), \mathbf{Y})$ is coming out of the joint source X, Y, $P_1 \to 0$ as $n \to \infty$
2. Note that $\mathbf{C}(M')$ and $\mathbf{Y}$ are independent and thus by Packing lemma,

$$P_2 \leq 2^{-N(I(X;Y)-R-3\epsilon)} \tag{1}$$

Since ϵ can be made arbitrarily small as N increase, as long as $I(X; Y) - 3\epsilon > R$, we can make P_2 arbitrarily small also given a sufficiently large N

A bit more caveat

- We *want* to show that there exists a code $\mathbf{c}^*(\cdot)$ such that $Pr(error|\mathbf{c}^*, m) \to 0$ no matter what message m is sent

A bit more caveat

- We *want* to show that there exists a code $\mathbf{c}^*(\cdot)$ such that $Pr(error|\mathbf{c}^*, m) \to 0$ no matter what message m is sent
- But we actually show that the *average* error over all random codes can be made arbitrarily small for any message m. That is, $Pr(error|m) = E[Pr(error|\mathbf{C}, m)] \to 0$

A bit more caveat

- We *want* to show that there exists a code $\mathbf{c}^*(\cdot)$ such that $Pr(error|\mathbf{c}^*, m) \to 0$ no matter what message m is sent
- But we actually show that the *average* error over all random codes can be made arbitrarily small for any message m. That is, $Pr(error|m) = E[Pr(error|\mathbf{C}, m)] \to 0$
- If $Pr(error|m) = E[Pr(error|\mathbf{C}, m)] \to 0, \forall m$, then $Pr(error) = E[Pr(error|\mathbf{C}, M)]$ average over all code and messages have to go to zero as well. Consequently, the best code (in terms of lowest error average error) should also have $Pr(error|\mathbf{c}^*) \triangleq \delta \to 0$

A bit more caveat

- We *want* to show that there exists a code $\mathbf{c}^*(\cdot)$ such that $Pr(error|\mathbf{c}^*, m) \to 0$ no matter what message m is sent
- But we actually show that the *average* error over all random codes can be made arbitrarily small for any message m. That is, $Pr(error|m) = E[Pr(error|\mathbf{C}, m)] \to 0$
- If $Pr(error|m) = E[Pr(error|\mathbf{C}, m)] \to 0, \forall m$, then $Pr(error) = E[Pr(error|\mathbf{C}, M)]$ average over all code and messages have to go to zero as well. Consequently, the best code (in terms of lowest error average error) should also have $Pr(error|\mathbf{c}^*) \triangleq \delta \to 0$
- Without loss of generality and for simplicity, assume that all messages are equally likely $Pr(error|\mathbf{c}^*) = \frac{1}{2^{NR}} \sum_m Pr(error|\mathbf{c}^*, m) = \delta$

A bit more caveat

- We *want* to show that there exists a code $\mathbf{c}^*(\cdot)$ such that $Pr(error|\mathbf{c}^*, m) \to 0$ no matter what message m is sent
- But we actually show that the *average* error over all random codes can be made arbitrarily small for any message m. That is, $Pr(error|m) = E[Pr(error|\mathbf{C}, m)] \to 0$
- If $Pr(error|m) = E[Pr(error|\mathbf{C}, m)] \to 0, \forall m$, then $Pr(error) = E[Pr(error|\mathbf{C}, M)]$ average over all code and messages have to go to zero as well. Consequently, the best code (in terms of lowest error average error) should also have $Pr(error|\mathbf{c}^*) \triangleq \delta \to 0$
- Without loss of generality and for simplicity, assume that all messages are equally likely $Pr(error|\mathbf{c}^*) = \frac{1}{2^{NR}} \sum_m Pr(error|\mathbf{c}^*, m) = \delta$
- If we discard worse half of the codewords, for remaining m, we have $Pr(error|\mathbf{c}^*, m) \leq 2Pr(error|\mathbf{c}^*) = 2\delta \to 0$ as $N \to \infty$

A bit more caveat

- We *want* to show that there exists a code $\mathbf{c}^*(\cdot)$ such that $Pr(error|\mathbf{c}^*, m) \to 0$ no matter what message m is sent
- But we actually show that the *average* error over all random codes can be made arbitrarily small for any message m. That is, $Pr(error|m) = E[Pr(error|\mathbf{C}, m)] \to 0$
- If $Pr(error|m) = E[Pr(error|\mathbf{C}, m)] \to 0, \forall m$, then $Pr(error) = E[Pr(error|\mathbf{C}, M)]$ average over all code and messages have to go to zero as well. Consequently, the best code (in terms of lowest error average error) should also have $Pr(error|\mathbf{c}^*) \triangleq \delta \to 0$
- Without loss of generality and for simplicity, assume that all messages are equally likely $Pr(error|\mathbf{c}^*) = \frac{1}{2^{NR}} \sum_m Pr(error|\mathbf{c}^*, m) = \delta$
- If we discard worse half of the codewords, for remaining m, we have $Pr(error|\mathbf{c}^*, m) \leq 2Pr(error|\mathbf{c}^*) = 2\delta \to 0$ as $N \to \infty$
- Even though the rate reduces from R to $R - \frac{1}{N}$ (number of messages from $2^{NR} \to 2^{NR-1}$). But we can still make the final rate arbitrarily close to the capacity as $N \to \infty$

Previously...

- Joint typical sequences
- Covering and Packing Lemmas
- Channel Coding Theorem
- Capacity of Gaussian channel
- Capacity of additive white Gaussian channel
- Forward proof of Channel Coding Theorem

This time

- Converse Proof of Channel Coding Theorem
- Non-white Gaussian Channel
- Rate-distortion problems
- Rate-distortion Theorem

Converse proof

We want to say that whenever the code rate is larger than the capacity, the probability of error will be non-zero

Converse proof

We want to say that whenever the code rate is larger than the capacity, the probability of error will be non-zero

Equivalently...

As long as the probability of error is 0, the rate of the code R has to be larger than the capacity

Converse proof

We want to say that whenever the code rate is larger than the capacity, the probability of error will be non-zero

Equivalently...

As long as the probability of error is 0, the rate of the code R has to be larger than the capacity

To continue the converse proof, we will need to introduce a simple result from Fano

Fano's inequality

Fano's inequality

Denote $Pr(error) = P_e = Pr(M \neq \hat{M})$, then $H(M|Y^N) \leq 1 + P_e H(M)$
Intuitively, if $P_e \to 0$, on average we will know M for certain given y and thus $\frac{1}{N} H(M|Y^N) \to 0$

Fano's inequality

Fano's inequality

Denote $Pr(error) = P_e = Pr(M \neq \hat{M})$, then $H(M|Y^N) \leq 1 + P_e H(M)$
Intuitively, if $P_e \to 0$, on average we will know M for certain given y and thus $\frac{1}{N} H(M|Y^N) \to 0$

Proof: Let $E = I(M \neq \hat{M})$, then

Fano's inequality

Fano's inequality

Denote $Pr(error) = P_e = Pr(M \neq \hat{M})$, then $H(M|Y^N) \leq 1 + P_e H(M)$
Intuitively, if $P_e \to 0$, on average we will know M for certain given y and thus $\frac{1}{N} H(M|Y^N) \to 0$

Proof: Let $E = I(M \neq \hat{M})$, then

$$H(M|Y^N) = H(M, E|Y^N) - H(E|Y^N, M)$$

Fano's inequality

Fano's inequality

Denote $Pr(error) = P_e = Pr(M \neq \hat{M})$, then $H(M|Y^N) \leq 1 + P_e H(M)$
Intuitively, if $P_e \to 0$, on average we will know M for certain given y and thus $\frac{1}{N} H(M|Y^N) \to 0$

Proof: Let $E = I(M \neq \hat{M})$, then

$$\begin{aligned} H(M|Y^N) &= H(M,E|Y^N) - H(E|Y^N, M) \\ &= H(M,E|Y^N) = H(E|Y^N) + H(M|Y^N, E) \end{aligned}$$

Fano's inequality

Fano's inequality

Denote $Pr(error) = P_e = Pr(M \neq \hat{M})$, then $H(M|Y^N) \leq 1 + P_e H(M)$
Intuitively, if $P_e \to 0$, on average we will know M for certain given y and thus $\frac{1}{N} H(M|Y^N) \to 0$

Proof: Let $E = I(M \neq \hat{M})$, then

$$\begin{aligned} H(M|Y^N) &= H(M, E|Y^N) - H(E|Y^N, M) \\ &= H(M, E|Y^N) = H(E|Y^N) + H(M|Y^N, E) \\ &\leq H(E) + H(M|Y^N, E) \end{aligned}$$

Fano's inequality

Fano's inequality

Denote $Pr(error) = P_e = Pr(M \neq \hat{M})$, then $H(M|Y^N) \leq 1 + P_e H(M)$
Intuitively, if $P_e \to 0$, on average we will know M for certain given y and thus $\frac{1}{N} H(M|Y^N) \to 0$

Proof: Let $E = I(M \neq \hat{M})$, then

$$\begin{aligned} H(M|Y^N) &= H(M, E|Y^N) - H(E|Y^N, M) \\ &= H(M, E|Y^N) = H(E|Y^N) + H(M|Y^N, E) \\ &\leq H(E) + H(M|Y^N, E) \\ &\leq 1 + P(E=0)H(M|Y^N, E=0) + P(E=1)H(M|Y^N, E=1) \end{aligned}$$

Fano's inequality

Fano's inequality

Denote $Pr(error) = P_e = Pr(M \neq \hat{M})$, then $H(M|Y^N) \leq 1 + P_e H(M)$
Intuitively, if $P_e \to 0$, on average we will know M for certain given y and thus $\frac{1}{N} H(M|Y^N) \to 0$

Proof: Let $E = I(M \neq \hat{M})$, then

$$
\begin{aligned}
H(M|Y^N) &= H(M, E|Y^N) - H(E|Y^N, M) \\
&= H(M, E|Y^N) = H(E|Y^N) + H(M|Y^N, E) \\
&\leq H(E) + H(M|Y^N, E) \\
&\leq 1 + P(E=0)H(M|Y^N, E=0) + P(E=1)H(M|Y^N, E=1) \\
&\leq 1 + 0 + P_e H(M|Y^N, E=1) \overset{(d)}{\leq} 1 + P_e H(M)
\end{aligned}
$$

Converse proof

$$R = \frac{H(M)}{N} = \frac{1}{N}\left[I(M;Y^N) + H(M|Y^N)\right]$$

Converse proof

$$R = \frac{H(M)}{N} = \frac{1}{N}\left[I(M;Y^N) + H(M|Y^N)\right] \leq \frac{1}{N}\left[I(X^N;Y^N) + H(M|Y^N)\right]$$

Converse proof

$$R = \frac{H(M)}{N} = \frac{1}{N}\left[I(M;Y^N) + H(M|Y^N)\right] \leq \frac{1}{N}\left[I(X^N;Y^N) + H(M|Y^N)\right]$$
$$= \frac{1}{N}\left[H(Y^N) - H(Y^N|X^N) + H(M|Y^N)\right]$$

Converse proof

$$R = \frac{H(M)}{N} = \frac{1}{N}\left[I(M;Y^N) + H(M|Y^N)\right] \leq \frac{1}{N}\left[I(X^N;Y^N) + H(M|Y^N)\right]$$
$$= \frac{1}{N}\left[H(Y^N) - H(Y^N|X^N) + H(M|Y^N)\right]$$
$$= \frac{1}{N}\left[H(Y^N) - \sum_i H(Y_i|X^N, Y^{i-1}) + H(M|Y^N)\right]$$

Converse proof

$$
\begin{aligned}
R = \frac{H(M)}{N} &= \frac{1}{N}\left[I(M;Y^N) + H(M|Y^N)\right] \le \frac{1}{N}\left[I(X^N;Y^N) + H(M|Y^N)\right] \\
&= \frac{1}{N}\left[H(Y^N) - H(Y^N|X^N) + H(M|Y^N)\right] \\
&= \frac{1}{N}\left[H(Y^N) - \sum_i H(Y_i|X^N, Y^{i-1}) + H(M|Y^N)\right] \\
&= \frac{1}{N}\left[H(Y^N) - \sum_i H(Y_i|X_i) + H(M|Y^N)\right]
\end{aligned}
$$

Converse proof

$$
\begin{aligned}
R = \frac{H(M)}{N} &= \frac{1}{N}\left[I(M;Y^N) + H(M|Y^N)\right] \leq \frac{1}{N}\left[I(X^N;Y^N) + H(M|Y^N)\right] \\
&= \frac{1}{N}\left[H(Y^N) - H(Y^N|X^N) + H(M|Y^N)\right] \\
&= \frac{1}{N}\left[H(Y^N) - \sum_i H(Y_i|X^N, Y^{i-1}) + H(M|Y^N)\right] \\
&= \frac{1}{N}\left[H(Y^N) - \sum_i H(Y_i|X_i) + H(M|Y^N)\right] \\
&\leq \frac{1}{N}\left[\sum_i H(Y_i) - \sum_i H(Y_i|X_i) + H(M|Y^N)\right]
\end{aligned}
$$

Converse proof

$$
\begin{aligned}
R = \frac{H(M)}{N} &= \frac{1}{N}\left[I(M;Y^N) + H(M|Y^N)\right] \leq \frac{1}{N}\left[I(X^N;Y^N) + H(M|Y^N)\right] \\
&= \frac{1}{N}\left[H(Y^N) - H(Y^N|X^N) + H(M|Y^N)\right] \\
&= \frac{1}{N}\left[H(Y^N) - \sum_i H(Y_i|X^N, Y^{i-1}) + H(M|Y^N)\right] \\
&= \frac{1}{N}\left[H(Y^N) - \sum_i H(Y_i|X_i) + H(M|Y^N)\right] \\
&\leq \frac{1}{N}\left[\sum_i H(Y_i) - \sum_i H(Y_i|X_i) + H(M|Y^N)\right] \\
&= \frac{1}{N}\left[\sum_i I(X_i;Y_i) + H(M|Y^N)\right] = I(X;Y) + \frac{H(M|Y^N)}{N}
\end{aligned}
$$

Converse proof

$$R = \frac{H(M)}{N} = \frac{1}{N}\left[I(M;Y^N) + H(M|Y^N)\right] \leq \frac{1}{N}\left[I(X^N;Y^N) + H(M|Y^N)\right]$$

$$= \frac{1}{N}\left[H(Y^N) - H(Y^N|X^N) + H(M|Y^N)\right]$$

$$= \frac{1}{N}\left[H(Y^N) - \sum_i H(Y_i|X^N, Y^{i-1}) + H(M|Y^N)\right]$$

$$= \frac{1}{N}\left[H(Y^N) - \sum_i H(Y_i|X_i) + H(M|Y^N)\right]$$

$$\leq \frac{1}{N}\left[\sum_i H(Y_i) - \sum_i H(Y_i|X_i) + H(M|Y^N)\right]$$

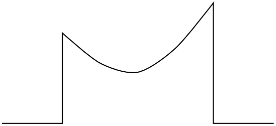
$$= \frac{1}{N}\left[\sum_i I(X_i;Y_i) + H(M|Y^N)\right] = I(X;Y) + \frac{H(M|Y^N)}{N} \to I(X;Y)$$

as $N \to \infty$ by Fano's inequality

Color channels

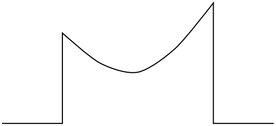
- We look into capacity of white Gaussian channel last time

Color channels



- We look into capacity of white Gaussian channel last time
- But sometimes noise power can be different for different band, consequently, “color” channels

Color channels



- We look into capacity of white Gaussian channel last time
- But sometimes noise power can be different for different band, consequently, "color" channels
- Intuitively, we should assign different amount of power to different band. Hence, we have an allocation problem

Color channels

- We look into capacity of white Gaussian channel last time
- But sometimes noise power can be different for different band, consequently, "color" channels
- Intuitively, we should assign different amount of power to different band. Hence, we have an allocation problem
- Without loss of generality, let's consider the discrete approximation, parallel Gaussian channel

Parallel Gaussian channels

- Consider that we have K parallel channels (K bands) and the corresponding noise powers are $\sigma_1^2, \sigma_2^2, \cdots, \sigma_K^2$

Parallel Gaussian channels

- Consider that we have K parallel channels (K bands) and the corresponding noise powers are $\sigma_1^2, \sigma_2^2, \cdots, \sigma_K^2$
- And say, we can allocate a total of P power to all channels. The powers assigned to the channels are $P_1, P_2, \cdots, P_K$. So we need $\sum_{i=1}^K P_i \leq P$

Parallel Gaussian channels

- Consider that we have K parallel channels (K bands) and the corresponding noise powers are $\sigma_1^2, \sigma_2^2, \cdots, \sigma_K^2$
- And say, we can allocate a total of P power to all channels. The powers assigned to the channels are $P_1, P_2, \cdots, P_K$. So we need $\sum_{i=1}^K P_i \leq P$
- Therefore, for the k-th channel, we can transmit $\frac{1}{2}\log\left(1+\frac{P_k}{\sigma_k^2}\right)$ bits per channel use

Parallel Gaussian channels

- Consider that we have K parallel channels (K bands) and the corresponding noise powers are $\sigma_1^2, \sigma_2^2, \cdots, \sigma_K^2$
- And say, we can allocate a total of P power to all channels. The powers assigned to the channels are $P_1, P_2, \cdots, P_K$. So we need $\sum_{i=1}^K P_i \leq P$
- Therefore, for the k-th channel, we can transmit $\frac{1}{2}\log\left(1+\frac{P_k}{\sigma_k^2}\right)$ bits per channel use
- So our goal is to assign $P_1, P_2, \cdots, P_K \geq 0$ ($\sum_{k=1}^K P_k \leq P$) such that the total capacity

$$\sum_{k=1}^{K} \frac{1}{2}\log\left(1+\frac{P_k}{\sigma_k^2}\right)$$

is maximize

KKT conditions

Let's list all the KKT conditions for the optimization problem

$$\max \sum_{k=1}^{K} \frac{1}{2} \log \left(1 + \frac{P_k}{\sigma_k^2}\right) \qquad \text{such that}$$

$$P_1, \cdots, P_K \geq 0, \qquad \sum_{k=1}^{K} P_k \leq P$$

$$\frac{\partial}{\partial P_i} \left[\sum_{k=1}^{K} \frac{1}{2} \log \left(1 + \frac{P_k}{\sigma_k^2}\right) + \sum_{k=1}^{K} \lambda_k P_k - \mu \left(\sum_{k=1}^{K} P_k - P \right) \right] = 0$$

KKT conditions

Let's list all the KKT conditions for the optimization problem

$$\max \sum_{k=1}^{K} \frac{1}{2} \log \left(1 + \frac{P_k}{\sigma_k^2}\right) \qquad \text{such that}$$

$$P_1, \cdots, P_K \geq 0, \qquad \sum_{k=1}^{K} P_k \leq P$$

$$\frac{\partial}{\partial P_i} \left[\sum_{k=1}^{K} \frac{1}{2} \log \left(1 + \frac{P_k}{\sigma_k^2}\right) + \sum_{k=1}^{K} \lambda_k P_k - \mu \left(\sum_{k=1}^{K} P_k - P \right) \right] = 0$$

$$\mu, \lambda_1, \cdots, \lambda_K \geq 0$$

KKT conditions

Let's list all the KKT conditions for the optimization problem

$$\max \sum_{k=1}^{K} \frac{1}{2} \log \left(1 + \frac{P_k}{\sigma_k^2}\right) \qquad \text{such that}$$

$$P_1, \cdots, P_K \geq 0, \qquad \sum_{k=1}^{K} P_k \leq P$$

$$\frac{\partial}{\partial P_i} \left[\sum_{k=1}^{K} \frac{1}{2} \log \left(1 + \frac{P_k}{\sigma_k^2}\right) + \sum_{k=1}^{K} \lambda_k P_k - \mu \left(\sum_{k=1}^{K} P_k - P \right) \right] = 0$$

$$\mu, \lambda_1, \cdots, \lambda_K \geq 0, P_1, \cdots, P_K \geq 0, \sum_{k=1}^{K} P_k \leq P$$

KKT conditions

Let's list all the KKT conditions for the optimization problem

$$\max \sum_{k=1}^{K} \frac{1}{2} \log\left(1+\frac{P_k}{\sigma_k^2}\right) \qquad \text{such that}$$

$$P_1, \cdots, P_K \geq 0, \qquad \sum_{k=1}^{K} P_k \leq P$$

$$\frac{\partial}{\partial P_i}\left[\sum_{k=1}^{K} \frac{1}{2} \log\left(1+\frac{P_k}{\sigma_k^2}\right) + \sum_{k=1}^{K} \lambda_k P_k - \mu\left(\sum_{k=1}^{K} P_k - P\right)\right] = 0$$

$$\mu, \lambda_1, \cdots, \lambda_K \geq 0, P_1, \cdots, P_K \geq 0, \sum_{k=1}^{K} P_k \leq P$$

$$\mu\left(\sum_{k=1}^{K} P_k - P\right) = 0, \qquad \lambda_k P_k = 0, \forall k$$

Capacity of parallel channels

$$\frac{\partial}{\partial P_i}\left[\sum_{k=1}^{K}\frac{1}{2}\log\left(1+\frac{P_k}{\sigma_k^2}\right)+\sum_{k=1}^{K}\lambda_k P_k-\mu\left(\sum_{k=1}^{K}P_k-P\right)\right]=0$$

Capacity of parallel channels

$$\frac{\partial}{\partial P_i}\left[\sum_{k=1}^{K}\frac{1}{2}\log\left(1+\frac{P_k}{\sigma_k^2}\right)+\sum_{k=1}^{K}\lambda_k P_k-\mu\left(\sum_{k=1}^{K}P_k-P\right)\right]=0$$

$$\Rightarrow\frac{1}{2}\frac{1}{P_i+\sigma_i^2}=\mu-\lambda_i$$

Capacity of parallel channels

$$\frac{\partial}{\partial P_i}\left[\sum_{k=1}^{K}\frac{1}{2}\log\left(1+\frac{P_k}{\sigma_k^2}\right)+\sum_{k=1}^{K}\lambda_k P_k-\mu\left(\sum_{k=1}^{K}P_k-P\right)\right]=0$$

$$\Rightarrow\frac{1}{2}\frac{1}{P_i+\sigma_i^2}=\mu-\lambda_i\Rightarrow P_i+\sigma_i^2=\frac{1}{2(\mu-\lambda_i)}$$

Capacity of parallel channels

$$\frac{\partial}{\partial P_i}\left[\sum_{k=1}^{K}\frac{1}{2}\log\left(1+\frac{P_k}{\sigma_k^2}\right)+\sum_{k=1}^{K}\lambda_k P_k-\mu\left(\sum_{k=1}^{K}P_k-P\right)\right]=0$$

$$\Rightarrow\frac{1}{2}\frac{1}{P_i+\sigma_i^2}=\mu-\lambda_i\Rightarrow P_i+\sigma_i^2=\frac{1}{2(\mu-\lambda_i)}$$

Since $\lambda_i P_i=0$, for $P_i>0$, we have $\lambda_i=0$ and thus

$$P_i+\sigma_i^2=\frac{1}{2\mu}$$

Capacity of parallel channels

$$\frac{\partial}{\partial P_i}\left[\sum_{k=1}^{K}\frac{1}{2}\log\left(1+\frac{P_k}{\sigma_k^2}\right)+\sum_{k=1}^{K}\lambda_k P_k-\mu\left(\sum_{k=1}^{K}P_k-P\right)\right]=0$$

$$\Rightarrow\frac{1}{2}\frac{1}{P_i+\sigma_i^2}=\mu-\lambda_i\Rightarrow P_i+\sigma_i^2=\frac{1}{2(\mu-\lambda_i)}$$

Since $\lambda_i P_i = 0$, for $P_i > 0$, we have $\lambda_i = 0$ and thus

$$P_i+\sigma_i^2=\frac{1}{2\mu}$$

This suggests that $\mu > 0$ and thus $\sum_{k=1}^{K} P_k = P$

Capacity of parallel channels

$$\frac{\partial}{\partial P_i}\left[\sum_{k=1}^{K}\frac{1}{2}\log\left(1+\frac{P_k}{\sigma_k^2}\right)+\sum_{k=1}^{K}\lambda_k P_k-\mu\left(\sum_{k=1}^{K}P_k-P\right)\right]=0$$

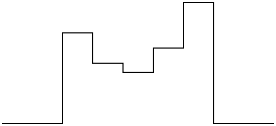
$$\Rightarrow\frac{1}{2}\frac{1}{P_i+\sigma_i^2}=\mu-\lambda_i\Rightarrow P_i+\sigma_i^2=\frac{1}{2(\mu-\lambda_i)}$$

Since $\lambda_i P_i = 0$, for $P_i > 0$, we have $\lambda_i = 0$ and thus

$$P_i+\sigma_i^2=\frac{1}{2\mu}=constant$$

This suggests that $\mu > 0$ and thus $\sum_{k=1}^{K} P_k = P$

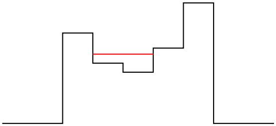
Water-filling interpretation



From $P_i + \sigma_i^2 = const$, power can be allocated intuitively as filling water to a pond (hence "water-filling")

Example

Water-filling interpretation

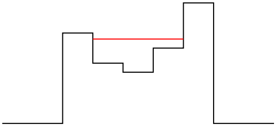


From $P_i + \sigma_i^2 = const$, power can be allocated intuitively as filling water to a pond (hence "water-filling")

Example

- $P_1 = 0, P_2 = 0.3, P_3 = 0.6, P_4 = 0, P_5 = 0$

Water-filling interpretation

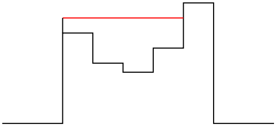


From $P_i + \sigma_i^2 = const$, power can be allocated intuitively as filling water to a pond (hence "water-filling")

Example

- $P_1 = 0, P_2 = 0.3, P_3 = 0.6, P_4 = 0, P_5 = 0$
- $P_1 = 0, P_2 = 0.8, P_3 = 1.1, P_4 = 0.3, P_5 = 0$

Water-filling interpretation



From $P_i + \sigma_i^2 = const$, power can be allocated intuitively as filling water to a pond (hence "water-filling")

Example

- $P_1 = 0, P_2 = 0.3, P_3 = 0.6, P_4 = 0, P_5 = 0$
- $P_1 = 0, P_2 = 0.8, P_3 = 1.1, P_4 = 0.3, P_5 = 0$
- $P_1 = 0.5, P_2 = 1.5, P_3 = 1.8, P_4 = 1, P_5 = 0$

Water-filling interpretation

From $P_i + \sigma_i^2 = const$, power can be allocated intuitively as filling water to a pond (hence "water-filling")

Example

- $P_1 = 0, P_2 = 0.3, P_3 = 0.6, P_4 = 0, P_5 = 0$
- $P_1 = 0, P_2 = 0.8, P_3 = 1.1, P_4 = 0.3, P_5 = 0$
- $P_1 = 0.5, P_2 = 1.5, P_3 = 1.8, P_4 = 1, P_5 = 0$