COMP2610 – Information Theory

Lecture 11: Entropy and Coding

Mark Reid and Aditya Menon

Research School of Computer Science
The Australian National University

THE AUSTRALIAN NATIONAL UNIVERSITY

August 26th, 2014

Brief Overview of Course (Next 6 Weeks)

- How can we quantify information? [Aditya]
  - Basic Definitions and Key Concepts
  - Probability, Entropy & Information
- How can we make good guesses? [Aditya]
  - Probabilistic Inference
  - Bayes Theorem
- How much redundancy can we safely remove? [Mark]
  - Compression
  - Source Coding Theorem, Kraft Inequality
  - Block, Huffman, and Lempel-Ziv Coding
- How much noise can we correct and how? [Mark]
  - Noisy-Channel Coding
  - Repetition Codes, Hamming Codes
- What is randomness? [Marcus]
  - Kolmogorov Complexity
  - Algorithmic Information Theory

Brief Overview of Course (Next 6 Weeks)

- How can we quantify information? [Aditya]
  - Basic Definitions and Key Concepts
  - Probability, Entropy & Information
- How can we make good guesses? [Aditya]
  - Probabilistic Inference
  - Bayes Theorem
- How much redundancy can we safely remove? [Mark]
  - Compression
  - Source Coding Theorem, Kraft Inequality
  - Block, Huffman, and Lempel-Ziv Coding
- How much noise can we correct and how? [Mark]
  - Noisy-Channel Coding
  - Repetition Codes, Hamming Codes
- What is randomness? [Marcus]
  - Kolmogorov Complexity
  - Algorithmic Information Theory

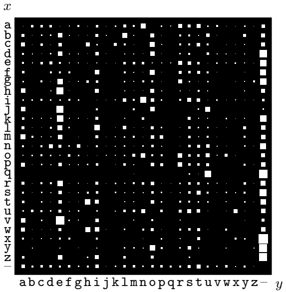
What is Compression?

Cn y rd ths mssg wtht ny vwls?

What is Compression?

Cn y rd ths mssg wtht ny vwls?

It is not too difficult to read as there is redundancy in English text.
(Estimates of 1-1.5 bits per character, compared to $\log_2 26 \approx 4.7$)



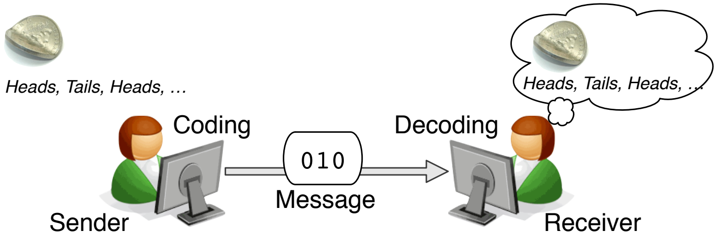
(a) $P(y \mid x)$

- If you see a "q", it is very likely to be followed with a "u"
- The letter "e" is much more common than "j"
- Compression exploits differences in relative probability of symbols or blocks of symbols

A General Communication Game

Data compression is the process of replacing a message with a smaller message which can be reliably converted back to the original.

- Sender & Receiver agree on code for each outcome ahead of time (e.g., 0 for *Heads*; 1 for *Tails*)
- Sender observes outcomes then codes and sends message
- Receiver decodes message and recovers outcome sequence
- Want small messages on average when outcomes are from a fixed, known, but uncertain source (e.g., coin flips with known bias)



What is the best we can do?

Consider a coin with $P(Heads) = 0.9$. If we want perfect transmission:

- Coding single outcomes requires 1 bit/outcome
- Coding 10 outcomes at a time needs 1 bits/outcome

What is the best we can do?

Consider a coin with $P(Heads) = 0.9$. If we want perfect transmission:

- Coding single outcomes requires 1 bit/outcome
- Coding 10 outcomes at a time needs 1 bits/outcome

However, if we are happy to fail on up to 2% of the sequences we can ignore any sequence of 10 outcomes with more than 3 tails (*Why?*).

What is the best we can do?

Consider a coin with $P(Heads) = 0.9$. If we want perfect transmission:

- Coding single outcomes requires 1 bit/outcome
- Coding 10 outcomes at a time needs 1 bits/outcome

However, if we are happy to fail on up to 2% of the sequences we can ignore any sequence of 10 outcomes with more than 3 tails (*Why?*). But there are only $176 < 2^8$ sequences with 3 or fewer tails (*Why?*).

- Coding 10 outcomes with 2% failure doable with 0.8 bits/outcome
- *Smallest bits/outcome needed for 10,000 outcome sequences?*

What is the best we can do?

Consider a coin with $P(Heads) = 0.9$. If we want perfect transmission:

- Coding single outcomes requires 1 bit/outcome
- Coding 10 outcomes at a time needs 1 bits/outcome

However, if we are happy to fail on up to 2% of the sequences we can ignore any sequence of 10 outcomes with more than 3 tails (*Why?*). But there are only $176 < 2^8$ sequences with 3 or fewer tails (*Why?*).

- Coding 10 outcomes with 2% failure doable with 0.8 bits/outcome
- *Smallest bits/outcome needed for 10,000 outcome sequences?*

Source Coding Theorem (Informal Statement)

If you want to uniformly code large sequences of outcomes with any degree of reliability from a random source then the average number of bits per outcome you will **need** is roughly equal to the entropy of that source.

What is the best we can do?

Consider a coin with $P(Heads) = 0.9$. If we want perfect transmission:

- Coding single outcomes requires 1 bit/outcome
- Coding 10 outcomes at a time needs 1 bits/outcome

However, if we are happy to fail on up to 2% of the sequences we can ignore any sequence of 10 outcomes with more than 3 tails (*Why?*). But there are only $176 < 2^8$ sequences with 3 or fewer tails (*Why?*).

- Coding 10 outcomes with 2% failure doable with 0.8 bits/outcome
- *Smallest bits/outcome needed for 10,000 outcome sequences?*

Source Coding Theorem (Informal Statement)

If you want to uniformly code large sequences of outcomes with any degree of reliability from a random source then the average number of bits per outcome you will **need** is roughly equal to the entropy of that source.

To define: "Uniformly code", "large sequences", "degree of reliability", "average number of bits per outcome", "roughly equal"

Entropy and Information

Ensemble

An **ensemble** X is a triple $(x, \mathcal{A}_X, \mathcal{P}_X)$; x is a **random variable** taking **values** in $\mathcal{A}_X = \{a_1, a_2, \ldots, a_I\}$ with **probabilities** $\mathcal{P}_X = \{p_1, p_2, \ldots, p_I\}$.

Information

The **information** in the observation that $x = a_i$ (in the ensemble X) is

$$h(a_i) = \log_2 \frac{1}{p_i} = -\log_2 p_i$$

Entropy

The **entropy** of an ensemble X is the average information

$$H(X) = \mathbb{E}[h(x)] = \sum_i p_i h(a_i) = \sum_i p_i \log_2 \frac{1}{p_i}$$

Let X be an ensemble with outcomes $\mathtt{h}$ for *heads* with probability 0.9 and $\mathtt{t}$ for *tails* with probability 0.1.

- The outcome set is $\mathcal{A}_X = \{\mathtt{h}, \mathtt{t}\}$
- The probabilities are $\mathcal{P}_X = \{p_{\mathtt{h}} = 0.9, p_{\mathtt{t}} = 0.1\}$

Let X be an ensemble with outcomes $\mathtt{h}$ for *heads* with probability 0.9 and $\mathtt{t}$ for *tails* with probability 0.1.

- The outcome set is $\mathcal{A}_X = \{\mathtt{h}, \mathtt{t}\}$
- The probabilities are $\mathcal{P}_X = \{p_{\mathtt{h}} = 0.9, p_{\mathtt{t}} = 0.1\}$

Information in observing heads $h(\mathtt{h}) = \log_2 \frac{1}{p_{\mathtt{h}}} = \log_2 \frac{10}{9} \approx 0.15$

Information in observing tails $h(\mathtt{t}) = \log_2 \frac{1}{p_{\mathtt{t}}} = \log_2 10 \approx 3.32$

Entropy $H(X) = p_{\mathtt{h}} h(p_{\mathtt{h}}) + p_{\mathtt{t}} h(p_{\mathtt{t}}) \approx 0.9 \times 0.15 + 0.1 \times 3.32 = 0.47$

Let X be an ensemble with outcomes $\mathtt{h}$ for *heads* with probability 0.9 and $\mathtt{t}$ for *tails* with probability 0.1.

- The outcome set is $\mathcal{A}_X = \{\mathtt{h}, \mathtt{t}\}$
- The probabilities are $\mathcal{P}_X = \{p_{\mathtt{h}} = 0.9, p_{\mathtt{t}} = 0.1\}$

Information in observing heads $h(\mathtt{h}) = \log_2 \frac{1}{p_{\mathtt{h}}} = \log_2 \frac{10}{9} \approx 0.15$

Information in observing tails $h(\mathtt{t}) = \log_2 \frac{1}{p_{\mathtt{t}}} = \log_2 10 \approx 3.32$

Entropy $H(X) = p_{\mathtt{h}} h(p_{\mathtt{h}}) + p_{\mathtt{t}} h(p_{\mathtt{t}}) \approx 0.9 \times 0.15 + 0.1 \times 3.32 = 0.47$

One can think of $h(x)$ as the surprise at learning outcome x. The entropy $H(X)$ is the expected amount of surprise for a draw from X.

What is a Code?

A source code is a process for assigning names to outcomes. The names are typically expressed by strings of binary symbols.

We will denote the set of all finite binary strings by

$$\mathcal{B} \stackrel{\text{def}}{=} \{0, 1, 00, 01, 10, \dots\}$$

Source Code

Given an ensemble X, the function $c : \mathcal{A}_X \to \mathcal{B}$ is a **source code** for X. The number of symbols in $c(x)$ is the **length** $l(x)$ of the codeword for x. The **extension** of c is defined by $c(x_1 \dots x_n) = c(x_1) \dots c(x_n)$

What is a Code?

A source code is a process for assigning names to outcomes. The names are typically expressed by strings of binary symbols.

We will denote the set of all finite binary strings by

$$\mathcal{B} \stackrel{\text{def}}{=} \{0, 1, 00, 01, 10, \ldots\}$$

Source Code

Given an ensemble X, the function $c : \mathcal{A}_X \to \mathcal{B}$ is a **source code** for X.
The number of symbols in $c(x)$ is the **length** $l(x)$ of the codeword for x.
The **extension** of c is defined by $c(x_1 \ldots x_n) = c(x_1) \ldots c(x_n)$

Example:

- The code c names outcomes from $\mathcal{A}_X = \{\mathtt{r}, \mathtt{g}, \mathtt{b}\}$ by $c(\mathtt{r}) = 00$, $c(\mathtt{g}) = 10$, $c(\mathtt{b}) = 11$
- The length of the codeword for each outcome is 2.
- The extension of c gives $c(\mathtt{rgrb}) = 00100011$

Types of Codes

Let X be an ensemble and $c : \mathcal{A}_X \to \mathcal{B}$ a code for X. We say c is a:

Uniform Code if $l(x)$ is the same for all $x \in \mathcal{A}_X$

Variable-Length Code otherwise

Types of Codes

Let X be an ensemble and $c : \mathcal{A}_X \to \mathcal{B}$ a code for X. We say c is a:

Uniform Code if $l(x)$ is the same for all $x \in \mathcal{A}_X$

Variable-Length Code otherwise

Another important criteria for codes is whether the original symbol x can be unambiguously determined given $c(x)$. We say c is a:

Lossless Code if for all $x_1, x_2 \in \mathcal{A}_X$ we have $x_1 \neq x_2$ implies $c(x_1) \neq c(x_2)$

Lossy Code otherwise

Types of Codes

Let X be an ensemble and $c : \mathcal{A}_X \to \mathcal{B}$ a code for X. We say c is a:

Uniform Code if $l(x)$ is the same for all $x \in \mathcal{A}_X$

Variable-Length Code otherwise

Another important criteria for codes is whether the original symbol x can be unambiguously determined given $c(x)$. We say c is a:

Lossless Code if for all $x_1, x_2 \in \mathcal{A}_X$ we have $x_1 \neq x_2$ implies $c(x_1) \neq c(x_2)$

Lossy Code otherwise

Examples: Let $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

1. $c(\mathtt{a}) = 00$, $c(\mathtt{b}) = 01$, $c(\mathtt{c}) = 10$, $c(\mathtt{d}) = 11$ is uniform lossless
2. $c(\mathtt{a}) = 0$, $c(\mathtt{b}) = 10$, $c(\mathtt{c}) = 110$, $c(\mathtt{d}) = 111$ is variable-length lossless
3. $c(\mathtt{a}) = 0$, $c(\mathtt{b}) = 0$, $c(\mathtt{c}) = 110$, $c(\mathtt{d}) = 111$ is variable-length lossy
4. $c(\mathtt{a}) = 00$, $c(\mathtt{b}) = 00$, $c(\mathtt{c}) = 10$, $c(\mathtt{d}) = 11$ is uniform lossy
5. $c(\mathtt{a}) = -$, $c(\mathtt{b}) = -$, $c(\mathtt{c}) = 10$, $c(\mathtt{d}) = 11$ is uniform lossy

Three colour ensemble with $\mathcal{A}_X = \{\mathtt{r}, \mathtt{g}, \mathtt{b}\}$ with $\mathtt{r}$ twice as likely as $\mathtt{b}$ or $\mathtt{g}$

- $p_{\mathtt{r}} = 0.5$ and $p_{\mathtt{g}} = p_{\mathtt{b}} = 0.25$.

Suppose we use the following **uniform lossless** code

$$c(\mathtt{r}) = 00;\ c(\mathtt{g}) = 10;\ \text{and}\ c(\mathtt{b}) = 11$$

For example $c(\mathtt{rrgbrbr}) = 00001011001100$ will have 14 bits.
On average, we will use $l(\mathtt{r})p_{\mathtt{r}} + l(\mathtt{g})p_{\mathtt{g}} + l(\mathtt{b})p_{\mathtt{b}} = 2$ bits per outcome

Lossless Coding

Example: Colours

Three colour ensemble with $\mathcal{A}_X = \{\mathtt{r}, \mathtt{g}, \mathtt{b}\}$ with $\mathtt{r}$ twice as likely as $\mathtt{b}$ or $\mathtt{g}$

- $p_{\mathtt{r}} = 0.5$ and $p_{\mathtt{g}} = p_{\mathtt{b}} = 0.25$.

Suppose we use the following **uniform lossless** code

$$c(\mathtt{r}) = 00;\ c(\mathtt{g}) = 10;\ \text{and}\ c(\mathtt{b}) = 11$$

For example $c(\mathtt{rrgbrbr}) = 00001011001100$ will have 14 bits.
On average, we will use $l(\mathtt{r})p_{\mathtt{r}} + l(\mathtt{g})p_{\mathtt{g}} + l(\mathtt{b})p_{\mathtt{b}} = 2$ bits per outcome
Uniform coding gives a crude measure of information:
the number of bits required to assign equal length codes to each symbol

Raw Bit Content

If X is an ensemble with outcome set $\mathcal{A}_X$ then its **raw bit content** is

$$H_0(X) = \log_2 |\mathcal{A}_X|.$$

Raw Bit Content

Raw Bit Content

If X is an ensemble with outcome set $\mathcal{A}_X$ then its **raw bit content** is

$$H_0(X) = \log_2 |\mathcal{A}_X|.$$

x	$c(x)$
a	000
b	001
c	010
d	011
e	100
f	101
g	110
h	111

Example:
This is a uniform encoding of outcomes in $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}\}$:

- Each outcome is encoded using $H_0(X) = 3$ bits
- The probabilities of the outcomes are ignored
- Same as assuming a uniform distribution

For the purposes of compression, the exact codes don't matter – just the number of bits used.

Lossy Coding

Example: Colours

Three colour ensemble with $\mathcal{A}_X = \{\mathtt{r}, \mathtt{g}, \mathtt{b}\}$

- $p_{\mathtt{r}} = 0.5$ and $p_{\mathtt{g}} = p_{\mathtt{b}} = 0.25$.

Using **uniform** **lossy** code:

- $c(\mathtt{r}) = 0$; $c(\mathtt{g}) = -$; and $c(\mathtt{b}) = 1$

Examples:

$c(\mathtt{rrrrrrr}) = 0000000$; $c(\mathtt{rrbbrbr}) = 0011010$; $c(\mathtt{rrgbrbr}) = -$

Lossy Coding

Example: Colours

Three colour ensemble with $\mathcal{A}_X = \{\mathtt{r}, \mathtt{g}, \mathtt{b}\}$

- $p_{\mathtt{r}} = 0.5$ and $p_{\mathtt{g}} = p_{\mathtt{b}} = 0.25$.

Using **uniform** **lossy** code:

- $c(\mathtt{r}) = 0$; $c(\mathtt{g}) = -$; and $c(\mathtt{b}) = 1$

Examples:

$c(\mathtt{rrrrrrr}) = 0000000$; $c(\mathtt{rrbbrbr}) = 0011010$; $c(\mathtt{rrgbrbr}) = -$

What is probability we can code a sequence of N outcomes?

$$P(x_1 \ldots x_N \text{ has no } \mathtt{g}) = P(x_1 \neq \mathtt{g}) \ldots P(x_N \neq \mathtt{g}) = (1 - p_{\mathtt{g}})^N$$

Three colour ensemble with $\mathcal{A}_X = \{\mathtt{r}, \mathtt{g}, \mathtt{b}\}$

- $p_{\mathtt{r}} = 0.5$ and $p_{\mathtt{g}} = p_{\mathtt{b}} = 0.25$.

Using **uniform lossy** code:

- $c(\mathtt{r}) = 0$; $c(\mathtt{g}) = -$; and $c(\mathtt{b}) = 1$

Examples:

$c(\mathtt{rrrrrrr}) = 0000000$; $c(\mathtt{rrbbrbr}) = 0011010$; $c(\mathtt{rrgbrbr}) = -$

What is probability we can code a sequence of N outcomes?

$$P(x_1 \ldots x_N \text{ has no } \mathtt{g}) = P(x_1 \neq \mathtt{g}) \ldots P(x_N \neq \mathtt{g}) = (1 - p_{\mathtt{g}})^N$$

Given we can code a sequence of length N, how many bits are expected?

$$\mathbb{E}[l(X_1) + \cdots + l(X_N) | X_1 \neq \mathtt{g}, \ldots, X_N \neq \mathtt{g}] = \sum_{n=1} \mathbb{E}[l(X_n) | X_n \neq \mathtt{g}]$$

$$= N \left(l(\mathtt{r}) p_{\mathtt{r}} + l(\mathtt{b}) p_{\mathtt{b}} \right) / (1 - p_{\mathtt{g}}) = N$$

since $l(p_{\mathtt{r}}) = l(p_{\mathtt{b}}) = 1$ and $p_{\mathtt{r}} + p_{\mathtt{b}} = 1 - p_{\mathtt{g}}$.

Three colour ensemble with $\mathcal{A}_X = \{\mathtt{r}, \mathtt{g}, \mathtt{b}\}$

- $p_\mathtt{r} = 0.5$ and $p_\mathtt{g} = p_\mathtt{b} = 0.25$.

Using **uniform lossy** code:

- $c(\mathtt{r}) = 0$; $c(\mathtt{g}) = -$; and $c(\mathtt{b}) = 1$

Examples:

$c(\mathtt{rrrrrrr}) = 0000000$; $c(\mathtt{rrbbrbr}) = 0011010$; $c(\mathtt{rrgbrbr}) = -$

What is probability we can code a sequence of N outcomes?

$$P(x_1 \ldots x_N \text{ has no } \mathtt{g}) = P(x_1 \neq \mathtt{g}) \ldots P(x_N \neq \mathtt{g}) = (1 - p_\mathtt{g})^N$$

Given we can code a sequence of length N, how many bits are expected?

$$\mathbb{E}[l(X_1) + \cdots + l(X_N) | X_1 \neq \mathtt{g}, \ldots, X_N \neq \mathtt{g}] = \sum_{n=1} \mathbb{E}[l(X_n) | X_n \neq \mathtt{g}]$$

$$= N \left(l(\mathtt{r}) p_\mathtt{r} + l(\mathtt{b}) p_\mathtt{b} \right) / (1 - p_\mathtt{g}) = N = N \log_2 |\{\mathtt{r}, \mathtt{b}\}|$$

since $I(p_\mathtt{r}) = I(p_\mathtt{b}) = 1$ and $p_\mathtt{r} + p_\mathtt{b} = 1 - p_\mathtt{g}$.

Essential Bit Content

There is an inherent trade off between the number of bits required in a uniform lossy code and the probability of being able to code an outcome

Smallest δ-sufficient subset

Let X be an ensemble and for $\delta \geq 0$ define S_δ to be the smallest subset of $\mathcal{A}_X$ such that

$$P(x \in S_\delta) \geq 1 - \delta$$

Essential Bit Content

There is an inherent trade off between the number of bits required in a uniform lossy code and the probability of being able to code an outcome

Smallest δ-sufficient subset

Let X be an ensemble and for $\delta \geq 0$ define S_δ to be the smallest subset of $\mathcal{A}_X$ such that

$$P(x \in S_\delta) \geq 1 - \delta$$

x	$P(\mathbf{x})$
a	1/4
b	1/4
c	1/4
d	3/16
e	1/64
f	1/64
g	1/64
h	1/64

- Outcomes ranked (high–low) by $P(x = a_i)$ removed to make set S_δ with $P(x \in S_\delta) \geq 1 - \delta$

  $\delta = 0 \ : \ S_\delta = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}\}$

Essential Bit Content

There is an inherent trade off between the number of bits required in a uniform lossy code and the probability of being able to code an outcome

Smallest δ-sufficient subset

Let X be an ensemble and for $\delta \geq 0$ define S_δ to be the smallest subset of $\mathcal{A}_X$ such that

$$P(x \in S_\delta) \geq 1 - \delta$$

x	$P(\mathbf{x})$
a	1/4
b	1/4
c	1/4
d	3/16
e	1/64
f	1/64
g	1/64

- Outcomes ranked (high–low) by $P(x = a_i)$ removed to make set S_δ with $P(x \in S_\delta) \geq 1 - \delta$

$$\delta = 0 \ : \ S_\delta = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}\}$$

$$\delta = 1/64 \ : \ S_\delta = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}\}$$

Essential Bit Content

There is an inherent trade off between the number of bits required in a uniform lossy code and the probability of being able to code an outcome

Smallest δ-sufficient subset

Let X be an ensemble and for $\delta \geq 0$ define S_δ to be the smallest subset of $\mathcal{A}_X$ such that

$$P(x \in S_\delta) \geq 1 - \delta$$

x	$P(\mathbf{x})$
a	1/4
b	1/4
c	1/4
d	3/16

- Outcomes ranked (high–low) by $P(x = a_i)$ removed to make set S_δ with $P(x \in S_\delta) \geq 1 - \delta$

$$\delta = 0 \ : \ S_\delta = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}\}$$

$$\delta = 1/64 \ : \ S_\delta = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}\}$$

$$\delta = 1/16 \ : \ S_\delta = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$$

Essential Bit Content

There is an inherent trade off between the number of bits required in a uniform lossy code and the probability of being able to code an outcome

Smallest δ-sufficient subset

Let X be an ensemble and for $\delta \geq 0$ define S_δ to be the smallest subset of $\mathcal{A}_X$ such that

$$P(x \in S_\delta) \geq 1 - \delta$$

x	$P(\mathbf{x})$
a	1/4

- Outcomes ranked (high–low) by $P(x = a_i)$ removed to make set S_δ with $P(x \in S_\delta) \geq 1 - \delta$

$$\delta = 0 \ : \ S_\delta = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}\}$$
$$\delta = 1/64 \ : \ S_\delta = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}\}$$
$$\delta = 1/16 \ : \ S_\delta = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$$
$$\delta = 3/4 \ : \ S_\delta = \{\mathtt{a}\}$$

Essential Bit Content

Trade off between a probability of δ of not coding an outcome and size of uniform code is captured by the essential bit content

Essential Bit Content

Let X be an ensemble then for $\delta \geq 0$ the **essential bit content** of X is

$$H_\delta(X) \stackrel{\text{def}}{=} \log_2 |S_\delta|$$

Essential Bit Content

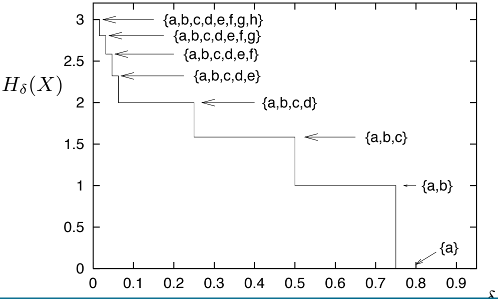
Trade off between a probability of δ of not coding an outcome and size of uniform code is captured by the essential bit content

Essential Bit Content

Let X be an ensemble then for $\delta \geq 0$ the **essential bit content** of X is

$$H_\delta(X) \stackrel{\text{def}}{=} \log_2 |S_\delta|$$

x	$P(\mathbf{x})$
a	1/4
b	1/4
c	1/4
d	3/16
e	1/64
f	1/64
g	1/64
h	1/64



The Source Coding Theorem for Uniform Codes

(Theorem 4.1 in MacKay)

Our aim this week is to understand this:

The Source Coding Theorem for Uniform Codes

Let X be an ensemble with entropy $H = H(X)$ bits. Given $\epsilon > 0$ and $0 < \delta < 1$, there exists a positive integer N_0 such that for all $N > N_0$

$$\left| \frac{1}{N} H_\delta \left(X^N \right) - H \right| < \epsilon.$$

The Source Coding Theorem for Uniform Codes

(Theorem 4.1 in MacKay)

Our aim this week is to understand this:

The Source Coding Theorem for Uniform Codes

Let X be an ensemble with entropy $H = H(X)$ bits. Given $\epsilon > 0$ and $0 < \delta < 1$, there exists a positive integer N_0 such that for all $N > N_0$

$$\left| \frac{1}{N} H_\delta \left(X^N \right) - H \right| < \epsilon.$$

What?

- The term $\frac{1}{N} H_\delta(X^N)$ is the average number of bits required to uniformly code all but a proportion δ of the symbols.
- Given a tiny probability of error δ, the average bits per symbol can be made as close to H as required.
- Even if we allow a large probability of error we cannot compress more than H bits ber symbol.