

COMP2610 – Information Theory

Lecture 11: Entropy and Coding

Assignment Project Exam Help

Robert C. Williamson

Research School of Computer Science

<https://powcoder.com>



Australian
National
University

Add WeChat powcoder

27 August 2018

Brief Recap of Course (Last 6 Weeks)

- How can we quantify information?
 - ▶ Basic Definitions and Key Concepts
 - ▶ Probability, Entropy & Information
- How can we make good guesses?
 - ▶ Probabilistic Inference
 - ▶ Bayes Theorem

- How much redundancy can we safely remove?
 - ▶ Compression
 - ▶ Source Coding Theorem, Kraft Inequality
 - ▶ Block, Huffman, and Lempel-Ziv Coding

- How much noise can we correct and how?
 - ▶ Noisy-Channel Coding
 - ▶ Repetition Codes, Hamming Codes
- What is randomness?
 - ▶ Kolmogorov Complexity
 - ▶ Algorithmic Information Theory

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Brief Overview of Course (Next 6 Weeks)

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

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Brief Overview of Course (Next 6 Weeks)

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

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

This time

Basic goal of compression

Key concepts: codes and their types, raw bit content, essential bit content

Assignment Project Exam Help

Informal statement of source coding theorem

<https://powcoder.com>

Add WeChat powcoder

1 Introduction

- Overview
- What is Compression?
- A Communication Game
- What's the best we can do?

2 Formalising Coding

- Entropy and Information: A Quick Review
- Defining Codes

3 Formalising Compression

- Reliability vs. Size
- Key Result: The Source Coding Theorem

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

What is Compression?

Cn y rd ths mssg wtht ny vwls?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

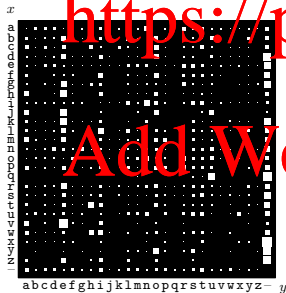
What is Compression?

Cn y rd ths mssg wtht ny vwls?

Assignment Project Exam Help

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) $P(y|x)$

<https://powcoder.com>

Add WeChat powcoder

- 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

Assignment Project Exam Help

Compression

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

<https://powcoder.com>

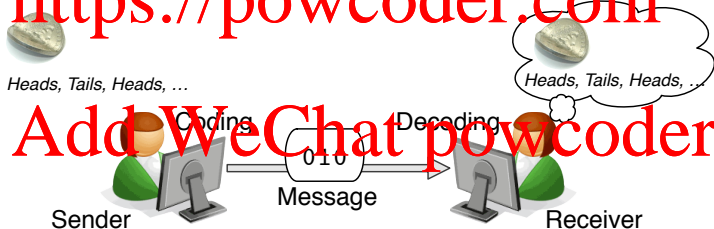
Add WeChat powcoder

A General Communication Game

Imagine the following game between Sender & Receiver:

- 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

<https://powcoder.com>



Goal: Want small messages **on average** when outcomes are from a **fixed, known, but uncertain** source (e.g., coin flips with known bias)

Assignment Project Exam Help

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

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

<https://powcoder.com>

Not very interesting!

Add WeChat powcoder

Sneak peek: source coding theorem

Assignment Project Exam Help

Consider a coin with $P(\text{Head}) = 0.9$. If we want perfect transmission:

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

Not very interesting!

<https://powcoder.com>

Things get interesting if we:

- accept errors in transmission
- allow variable length messages

Add WeChat powcoder

Sneak peek: source coding theorem

Assignment Project Exam Help

Consider a coin with $P(\text{Head}) = 0.9$. If we want perfect transmission:

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

Not very interesting!

<https://powcoder.com>

Things get interesting if we:

- accept errors in transmission (this week)
- allow variable length messages (next week)

Add WeChat powcoder

Sneak peek: source coding theorem

If we are happy to fail on up to 3% of the sequences we can ignore any sequence of 10 outcomes with more than 3 tails

Why? The number of tails follows a Binomial(10, 0.1) distribution

<https://powcoder.com>

Add WeChat powcoder

Sneak peek: source coding theorem

If we are happy to fail on up to 3% of the sequences we can ignore any sequence of 10 outcomes with more than 3 tails

Why? The number of tails follows a Binomial(10, 0.1) distribution

There are only $76 < 2^8$ sequences with 3 or fewer tails

So, we can just code those, and **ignore** the rest!

- Coding 10 outcomes with 2% failure doable with 8 bits, or 0.8 bits/outcome
- *Smallest bits/outcome needed for 10,000 outcome sequences?*

Generalisation: Source Coding Theorem

What happens when we generalise to arbitrary error probability, and sequence size?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Generalisation: Source Coding Theorem

What happens when we generalise to arbitrary error probability, and sequence size?

Assignment Project Exam Help

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.

<https://powcoder.com>
Add WeChat powcoder

Generalisation: Source Coding Theorem

What happens when we generalise to arbitrary error probability, and sequence size?

Assignment Project Exam Help

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”

1

Introduction

- Overview
- What is Compression?
- All Communication Same
- What's the best we can do?

Assignment Project Exam Help

2

Formalising Coding

- Entropy and Information: A Quick Review
- Defining Codes

<https://powcoder.com>

Add WeChat powcoder

3

Formalising Compression

- Reliability vs. Size
- Key Result: The Source Coding Theorem

Entropy and Information: Recap

Ensemble

An **ensemble** X is a triple $(x, \mathcal{A}_X, \mathcal{P}_X)$; x is a **random variable** taking

values in $\Omega_X = \{a_1, a_2, \dots, a_l\}$ with probabilities $\mathcal{P}_X = \{p_1, p_2, \dots, p_l\}$

<https://powcoder.com>

Add WeChat powcoder

Entropy and Information: Recap

Ensemble

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

Information

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

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

Add WeChat powcoder

Entropy and Information: Recap

Ensemble

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

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

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

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**.

Assignment Project Exam Help

We will denote the set of all finite binary strings by

$$\{0, 1\}^+ \stackrel{\text{def}}{=} \{0, 1, 00, 01, 10, \dots\}$$

Source Code

Given an ensemble X , the function $c : \mathcal{A}_X \rightarrow \{0, 1\}^+$ 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 $c(x_1 \dots x_n) = c(x_1) \dots c(x_n)$.

Add WeChat powcoder

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**.

Assignment Project Exam Help

We will denote the set of all finite binary strings by

$$\{0, 1\}^+ \stackrel{\text{def}}{=} \{0, 1, 00, 01, 10, \dots\}$$

Source Code

Given an ensemble X , the function $c : \mathcal{A}_X \rightarrow \{0, 1\}^+$ 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 $c(x_1 \dots x_n) = c(x_1) \dots c(x_n)$.

Example:

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

Types of Codes

Let X be an ensemble and $c: \mathcal{A}_X \rightarrow \{0, 1\}^+$ 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

<https://powcoder.com>

Add WeChat powcoder

Types of Codes

Let X be an ensemble and $c: \mathcal{A}_X \rightarrow \{0, 1\}^+$ 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

<https://powcoder.com>

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

Examples

Examples: Let $A_X = \{a, b, c, d\}$

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

A Note on Lossy Codes & Missing Codewords

When talking about a uniform lossy code c for $\mathcal{A}_X = \{a, b, c\}$ we write

Assignment Project Exam Help
 $c(a) = 0 \quad c(b) = 1 \quad c(c) = -$

where the symbol $-$ means “no codeword”. This is shorthand for “the receiver will decode this codeword incorrectly”

<https://powcoder.com>
For the purposes of these lectures, this is equivalent to the code

$$c(a) = 0 \quad c(b) = 1 \quad c(c) = 1$$

Add WeChat powcoder
and the sender and receiver agreeing that the codeword 1 should always be decoded as b

Assigning the outcome a_i the missing codeword “-” just means “it is not possible to send a_i reliably”

1

Introduction

- Overview
- What is Compression?
- All Compression is the Same
- What's the best we can do?

Assignment Project Exam Help

2

Formalising Coding

- Entropy and Information: A Quick Review
- Defining Codes

<https://powcoder.com>

3

Formalising Compression

- Reliability vs. Size
- Key Result: The Source Coding Theorem

Add WeChat powcoder

Lossless Coding

Example: Colours



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

- $p_r = 0.5$ and $p_g = p_b = 0.25$.

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

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

For example $c(rrgbrrr) = 00001011001100$ will have 14 bits.

On average, we will use $l(r)p_r + l(g)p_g + l(b)p_b = 2$ bits per outcome

- $2N$ bits to code a sequence of N outcomes

Raw Bit Content

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

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

<https://powcoder.com>

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

$$\mathcal{A}_X = \{a, b, c, d, e, f, g, h\}.$$

- 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 $\mathcal{X} = \{r, g, b\}$

- $p_r = 0.5$ and $p_g = p_b = 0.25$.

Using **uniform lossy** code:

- $c(r) = 0$, $c(g) = -$; and $c(b) = 1$

Examples:

$c(rrrrrrrr) = 0000000$; $c(rrbbrbr) = 0011010$; $c(rrgbrbr) = -$

Add WeChat powcoder

Lossy Coding

Example: Colours



Three colour ensemble with $\mathcal{X} = \{r, g, b\}$

- $p_r = 0.5$ and $p_g = p_b = 0.25$.

Using **uniform lossy** code:

- $c(r) = 0$, $c(g) = -$ and $c(b) = 1$

Examples:

$c(\text{rrrrrrrr}) = 0000000$; $c(\text{rrbbrrbr}) = 0011010$; $c(\text{rrgbrbr}) = -$

What is **probably** we can reliably code a sequence of N outcomes?

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

Lossy Coding

Example: Colours

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

Assignment Project Exam Help

$$P(x_1 \dots x_N \text{ has no } g) = P(x_1 \neq g) \dots P(x_N \neq g) = (1 - p_g)^N$$

<https://powcoder.com>

Add WeChat powcoder

Lossy Coding

Example: Colours

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

$$P(x_1 \dots x_N \text{ has no } g) = P(x_1 \neq g) \dots P(x_N \neq g) = (1 - p_g)^N$$

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

$$\begin{aligned} \mathbb{E}[I(X_1) + \dots + I(X_N) | X_1 \neq g, \dots, X_N \neq g] &= \sum_{n=1}^N \mathbb{E}[I(X_n) | X_n \neq g] \\ &= N(I(p_r)p_r + I(p_b)p_b) / (1 - p_g) = 2N \end{aligned}$$

since $I(p_r) = I(p_b) = 1$ and $p_r + p_b = 1 - p_g$.

- c.f. $2N$ bits with lossless code

Lossy Coding

Example: Colours

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

$$P(x_1 \dots x_N \text{ has no } g) = P(x_1 \neq g) \dots P(x_N \neq g) = (1 - p_g)^N$$

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

$$\begin{aligned} \mathbb{E}[I(X_1) + \dots + I(X_N) | X_1 \neq g, \dots, X_N \neq g] &= \sum_{n=1}^N \mathbb{E}[I(X_n) | X_n \neq g] \\ &= N(I(p_r)p_r + I(p_b)p_b) / (1 - p_g) = 2N \end{aligned}$$

since $I(p_r) = I(p_b) = 1$ and $p_r + p_b = 1 - p_g$.

- c.f. $2N$ bits with lossless code

Lossy Coding

Example: Colours

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

$$P(x_1 \dots x_N \text{ has no } g) = P(x_1 \neq g) \dots P(x_N \neq g) = (1 - p_g)^N$$

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

$$\begin{aligned} \mathbb{E}[I(X_1) + \dots + I(X_N) | X_1 \neq g, \dots, X_N \neq g] &= \sum_{n=1}^N \mathbb{E}[I(X_n) | X_n \neq g] \\ &= N(I(p_r) + I(p_b)) / (1 - p_g) = N \log_2 \{ \frac{1}{1 - p_g} \} \end{aligned}$$

since $I(p_r) = I(p_b) = 1$ and $p_r + p_b = 1 - p_g$.

- c.f. $2N$ bits with lossless code

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 $0 \leq \delta \leq 1$, define S_δ to be the smallest subset of \mathcal{X} such that

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

For small δ , **smallest** collection of **most likely** outcomes

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 $0 \leq \delta \leq 1$, define S_δ to be the smallest subset of \mathcal{X} such that

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

For small δ , **smallest** collection of **most likely** outcomes

If we uniformly code elements in S_δ , and ignore all others:

- We can code a sequence of length N with probability $(1 - \delta)^N$
- If we can code a sequence, its expected length is $N \log_2 |S_\delta|$

Essential Bit Content

Example

Intuitively, construct S_δ by removing elements of X in ascending order of probability, till we have reached the $1 - \delta$ threshold

x	$P(x)$
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 $P(x \in S_\delta) \geq 1 - \delta$
 $\delta = 0 : S_\delta = \{a, b, c, d, e, f, g, h\}$

Essential Bit Content

Example

Intuitively, construct S_δ by removing elements of X in ascending order of probability, till we have reached the $1 - \delta$ threshold

x	$P(x)$
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 $P(x \in S_\delta) \geq 1 - \delta$
- $\delta = 0 : S_\delta = \{a, b, c, d, e, f, g\}$
 $\delta = 1/64 : S_\delta = \{a, b, c, d, e, f, g\}$

Add WeChat powcoder

Essential Bit Content

Example

Intuitively, construct S_δ by removing elements of X in ascending order of probability, till we have reached the $1 - \delta$ threshold

x	$P(x)$
-----	--------

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

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

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

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

Add WeChat powcoder

Essential Bit Content

Example

Intuitively, construct S_δ by removing elements of X in ascending order of probability, till we have reached the $1 - \delta$ threshold

x	$P(x)$
a	1/4

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

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

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

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

$$\delta = 3/4 : S_\delta = \{a\}$$

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

For an ensemble X and $0 \leq \delta \leq 1$, the **essential bit content** of X is

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

<https://powcoder.com>

Add WeChat powcoder

Essential Bit Content

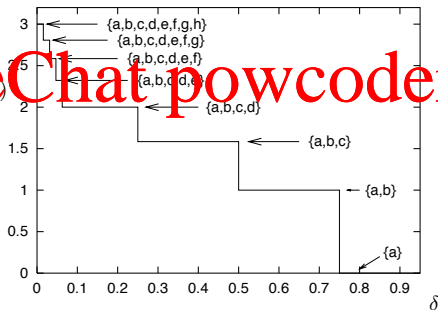
Trade off between a probability of δ of not coding an outcome and size of uniform code is captured by the **essential bit content**

Essential Bit Content

For an ensemble X and $0 \leq \delta \leq 1$, the **essential bit content** of X is

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

x	$P(x)$
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 next time 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 $N > N_0$

<https://powcoder.com>

Add WeChat powcoder

The Source Coding Theorem for Uniform Codes

(Theorem 4.1 in MacKay)

Our aim next time 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 $N > N_0$

<https://powcoder.com>

What?

- The term $\frac{1}{N} H(X^N)$ 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 per symbol.

Some Intuition for the SCT

Assignment Project Exam Help

- Don't code individual symbols in an ensemble; rather consider sequences of length N .

<https://powcoder.com>

- As length of sequence increases, the probability of seeing a “typical” sequence becomes much larger than “atypical” sequences.

Add WeChat powcoder

- Thus, we can get by with essentially assigning a unique codeword to each typical sequence

Next time

Recap: typical sets

Formal statement of source coding theorem

Proof of source coding theorem

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder