Winning *Guess Who?*: A Friendly Introduction to Information Theory

Ellen Zhang, Nathan Sheffield

IAP 2024

GUESS WHO?

Figure: MIT Math Department Edition

- Is your person in Applied Mathematics?
- Does your person's name come alphabetically after 'insert name'?

Guess Who?

What if you know your opponent favors certain characters, and are more likely to choose them?

Figure: How many questions do I need to ask for this distribution?

Entropy

Entropy gives us a way to quantify the minimal number of questions on average.

- Given random variable X with probability mass function $p(x)$,
$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x)$$
- Entropy measures the **minimal** amount of information required to describe X, in bits.
- Each bit is like a question.

ENTROPY

What is the entropy of professor X given the distribution

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

$$= -\left(4 \cdot \frac{2}{100} \log \frac{2}{100} + \frac{42}{100} \log \frac{42}{100} + \frac{50}{100} \log \frac{50}{100}\right) \approx 1.48 \text{ bits}$$

Interpretation: I need to ask at least 1.48 questions on average.

KRAFT INEQUALITY

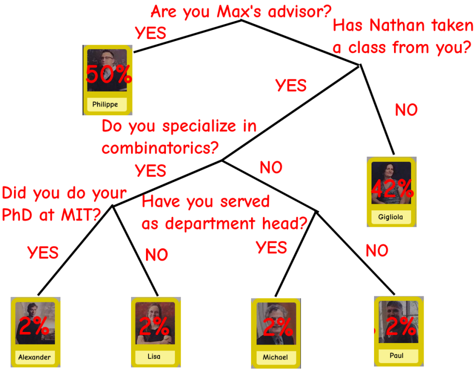


Figure: *Guess Who?* strategy visualized as a tree

Kraft Inequality

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

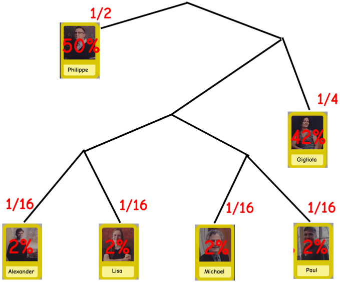


Figure: $\sum_{x \in \mathcal{X}} 2^{-l(x)} = 1/2 + 1/4 + 1/16 + 1/16 + 1/16 + 1/16 = 1$

Kraft Inequality

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

Proof.

- Walk down tree, choosing uniform random child each time
- Probability of ending on a given person x is $2^{-l(x)}$
- These events are disjoint

□

Kraft Inequality

For integer function l achieving $\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1$, there's a corresponding guessing strategy.

Proof.

Embed into the tree greedily from the top down. $\square$

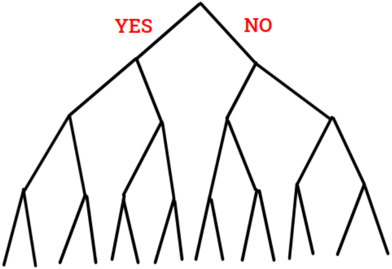
Kraft Inequality

For integer function l achieving $\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1$, there's a corresponding guessing strategy.

Proof.

Embed into the tree greedily from the top down. □

$$\{1, 3, 3, 3, 4, 4\}$$



Kraft Inequality

For integer function l achieving $\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1$, there's a corresponding guessing strategy.

Proof.

Embed into the tree greedily from the top down. □

$$\{3, 3, 3, 4, 4\}$$

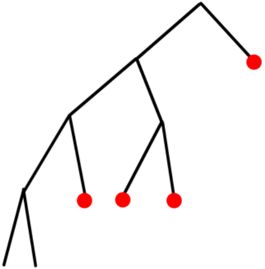
Kraft Inequality

For integer function l achieving $\sum_{x\in\mathcal{X}} 2^{-l(x)} \leq 1$, there's a corresponding guessing strategy.

Proof.

Embed into the tree greedily from the top down. □

$\{4, 4\}$

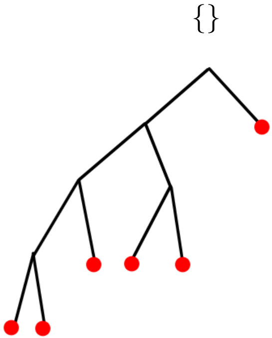


Kraft Inequality

For integer function l achieving $\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1$, there's a corresponding guessing strategy.

Proof.

Embed into the tree greedily from the top down. □



Shannon Coding

Goal: Choose integers l_i to minimize $\sum p_i l_i$, given $\sum 2^{-l_i} \leq 1$.

Shannon Coding

Goal: Choose integers l_i to minimize $\sum p_i l_i$, given $\sum 2^{-l_i} \leq 1$.

Lower Bound: Remove integrality constraint; use Lagrange multipliers.

$$\nabla \left(\sum p_i l_i \right) = \lambda \nabla \left(\sum 2^{-l_i} \right)$$

Shannon Coding

Goal: Choose integers l_i to minimize $\sum p_i l_i$, given $\sum 2^{-l_i} \leq 1$.

Lower Bound: Remove integrality constraint; use Lagrange multipliers.

$$\nabla \left(\sum p_i l_i \right) = \lambda \nabla \left(\sum 2^{-l_i} \right)$$

$$p_i = -\lambda \ln(2) 2^{-l_i}$$

$$\lambda = -1/\ln(2), \qquad p_i = 2^{-l_i}$$

Shannon Coding

Goal: Choose integers l_i to minimize $\sum p_i l_i$, given $\sum 2^{-l_i} \leq 1$.

Lower Bound: Remove integrality constraint; use Lagrange multipliers.

$$\nabla \left(\sum p_i l_i \right) = \lambda \nabla \left(\sum 2^{-l_i} \right)$$

$$p_i = -\lambda \ln(2) 2^{-l_i}$$

$$\lambda = -1/\ln(2), \qquad p_i = 2^{-l_i}$$

$$\sum p_i l_i = \sum p_i(-\log p_i) = H(X)$$

Shannon Coding

Goal: Choose integers l_i to minimize $\sum p_i l_i$, given $\sum 2^{-l_i} \leq 1$.

Lower Bound: Remove integrality constraint; use Lagrange multipliers.

$$\nabla \left(\sum p_i l_i \right) = \lambda \nabla \left(\sum 2^{-l_i} \right)$$

$$p_i = -\lambda \ln(2) 2^{-l_i}$$

$$\lambda = -1/\ln(2), \qquad p_i = 2^{-l_i}$$

$$\sum p_i l_i = \sum p_i(-\log p_i) = H(X)$$

Upper Bound: Round up; $l_i = \lceil -\log p_i \rceil$.

Shannon Coding

Goal: Choose integers l_i to minimize $\sum p_i l_i$, given $\sum 2^{-l_i} \leq 1$.

Lower Bound: Remove integrality constraint; use Lagrange multipliers.

$$\nabla \left(\sum p_i l_i \right) = \lambda \nabla \left(\sum 2^{-l_i} \right)$$

$$p_i = -\lambda \ln(2) 2^{-l_i}$$

$$\lambda = -1/\ln(2), \qquad p_i = 2^{-l_i}$$

$$\sum p_i l_i = \sum p_i(-\log p_i) = H(X)$$

Upper Bound: Round up; $l_i = \lceil -\log p_i \rceil$.

$$\sum p_i l_i = \sum p_i \lceil -\log p_i \rceil \leq 1 + \sum p_i(-\log p_i) = H(X) + 1$$

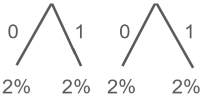
HUFFMAN CODING

An efficient way to assign binary codes to each outcome of X by continuously combining the two smallest probabilities.

$$2\%, 2\%, 4\%, 42\%, 50\%$$

Huffman Coding

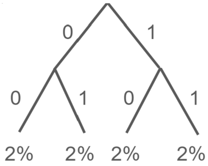
An efficient way to assign binary codes to each outcome of X by continuously combining the two smallest probabilities.



$$4\%, 4\%, 42\%, 50\%$$

HUFFMAN CODING

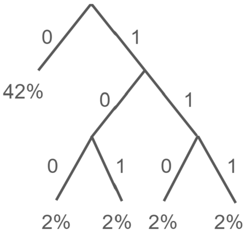
An efficient way to assign binary codes to each outcome of X by continuously combining the two smallest probabilities.



$$8\%, 42\%, 50\%$$

Huffman Coding

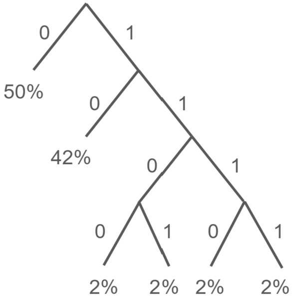
An efficient way to assign binary codes to each outcome of X by continuously combining the two smallest probabilities.



$$50\%, 50\%$$

HUFFMAN CODING

An efficient way to assign binary codes to each outcome of X by continuously combining the two smallest probabilities.



The average length of the code is $\sum p_i l_i \approx 1.66$ bits which is close to the entropy $H(X) = 1.48$ bits.

Huffman Coding

Shannon's Source Coding Theorem: Entropy $H(X)$ is the minimal average length that is theoretically possible. Huffman Coding is very close to this limit.

Theorem

Huffman Coding is optimal. That is, the average length $\sum_i p_i l_i$ *is minimal relative to all other codes.*

Canonical Codes

Assume that X takes on m discrete values, and that $p_1 \geq p_2 \geq \dots \geq p_m$.

Lemma

*There exists an optimal code, called a **cananical code**, that satisfies the following properties:*

1. *The lengths are ordered inversely with the probabilities (i.e., if $p_j > p_k$ then $l_j \leq l_k$.*
2. *The two longest codewords have the same length.*
3. *Two of the longest codewords differ only in the last bit and correspond to the two least likely symbols.*

Canonical Codes

Lemma (Part 1)

The lengths are ordered inversely with the probabilities (i.e., if $p_j > p_k$ then $l_j \leq l_k$.

An optimal code minimizes average length $\sum_i p_i l_i$.

If $p_j > p_k$ but $l_j > l_k$, then it is not optimal since we can swap the codewords and achieve a lower average length.

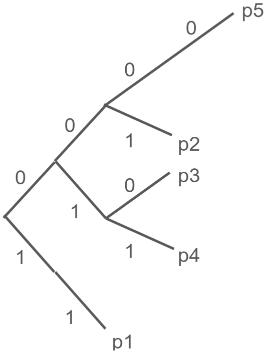
Thus the lengths must be ordered inversely with the probabilities for an optimal code.

Canonical Codes

Lemma (Part 2)

The two longest codewords have the same length.

Consider a possible tree

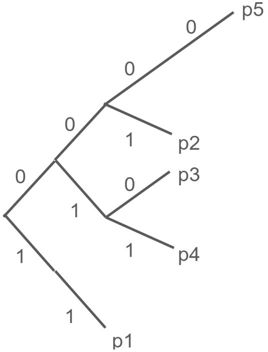


CANONICAL CODES

Lemma (Part 2)

The two longest codewords have the same length.

Consider a possible tree



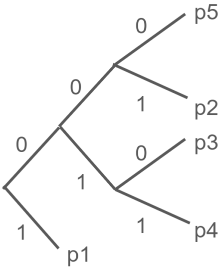
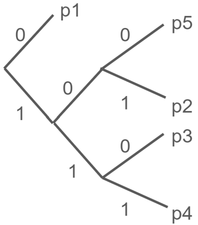


Figure: Trimming

Canonical Codes

Lemma (Part 3)

Two of the longest codewords differ only in the last bit and correspond to the two least likely symbols.



Canonical Codes

Lemma (Part 3)

Two of the longest codewords differ only in the last bit and correspond to the two least likely symbols.

Figure: Swapping

HUFFAMN CODE OPTIMALITY

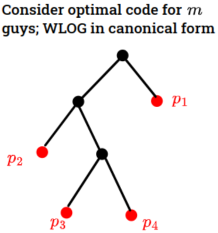
Huffman Code achieves minimum expected length.

- Assume that Huffman Coding is optimal for any distribution on $m - 1$ values.
- Consider any distribution on m values ordered so that $p_1 \geq p_2 \geq ... \geq p_m$.

HUFFAMN CODE OPTIMALITY

Huffman Code achieves minimum expected length.

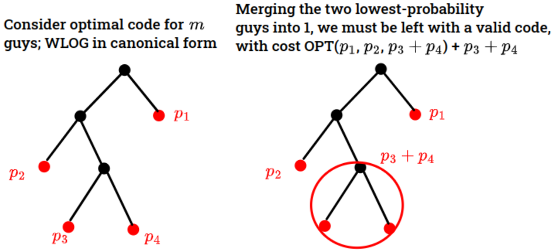
- Assume that Huffman Coding is optimal for any distribution on $m-1$ values.
- Consider any distribution on m values ordered so that $p_1 \geq p_2 \geq \ldots \geq p_m$.



HUFFAMN CODE OPTIMALITY

Huffman Code achieves minimum expected length.

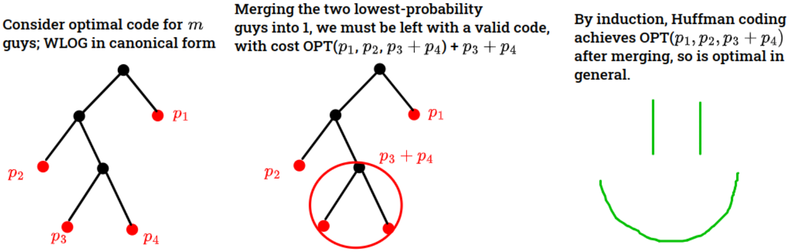
- Assume that Huffman Coding is optimal for any distribution on $m - 1$ values.
- Consider any distribution on m values ordered so that $p_1 \geq p_2 \geq \ldots \geq p_m$.



HUFFAMN CODE OPTIMALITY

Huffman Code achieves minimum expected length.

- Assume that Huffman Coding is optimal for any distribution on $m - 1$ values.
- Consider any distribution on m values ordered so that $p_1 \geq p_2 \geq \ldots \geq p_m$.



KL DIVERGENCE

This is all assuming that I know my opponent's true distribution $p(x)$. What if I believed it was $q(x)$? Then my expected length under Shannon Coding is not optimal.

KL Divergence

$$D(p||q) = \mathbb{E}_{x \sim p} \log \frac{p(x)}{q(x)}$$

KL Divergence

$$D(p||q) = \mathbb{E}_{x \sim p} \log \frac{p(x)}{q(x)}$$

What I thought your distribution was:

What it actually was:

20% 20% 15% 15% 15% 15%

$$D(p||q) = .2 \log(.2/.5) + .2 \log(.2/.42)$$
$$+ 4 * .15 \log(.15/.02)$$
$$= 1.27 \text{ bits}$$

KL Divergence

True distribution is $p(x)$ but I construct Shannon Code using $q(x)$:

$$H(p) + D(p||q) \leq \mathbb{E}_p l(x) < H(p) + D(p||q) + 1$$

KL Divergence

$$H(p) + D(p||q) \leq \mathbb{E}_p l(x) < H(p) + D(p||q) + 1$$

Proof.

$$\mathbb{E}_p l(x) = \sum_x p(x) \left\lceil \log \frac{1}{q(x)} \right\rceil < 1 + \sum_x p(x) \log \frac{1}{q(x)}$$

□

KL Divergence

$$H(p) + D(p||q) \leq \mathbb{E}_p l(x) < H(p) + D(p||q) + 1$$

Proof.

$$\begin{aligned}
\mathbb{E}_p l(x) &= \sum_x p(x) \left\lceil \log \frac{1}{q(x)} \right\rceil < 1 + \sum_x p(x) \log \frac{1}{q(x)} \\
&= \sum_x p(x) \log \frac{p(x)}{q(x)} + \sum_x p(x) \log \frac{1}{p(x)} + 1
\end{aligned}$$

□

KL Divergence

$$H(p) + D(p||q) \leq \mathbb{E}_p l(x) < H(p) + D(p||q) + 1$$

Proof.

$$\begin{aligned}
\mathbb{E}_p l(x) &= \sum_x p(x) \left\lceil \log \frac{1}{q(x)} \right\rceil < 1 + \sum_x p(x) \log \frac{1}{q(x)} \\
&= \sum_x p(x) \log \frac{p(x)}{q(x)} + \sum_x p(x) \log \frac{1}{p(x)} + 1 \\
&= D(p||q) + H(p) + 1
\end{aligned}$$

□

GAME THEORY [NICA, 2016]

Suppose you're playing against another person. You have n possibilities remaining for their character, and they have m remaining for yours. You really want to beat them.

Game Theory [Nica, 2016]

Suppose you're playing against another person. You have n possibilities remaining for their character, and they have m remaining for yours. You really want to beat them.

- If $\lceil \log_2(m) \rceil < \log_2(n)$, you should take a risk and try to eliminate all but $2^{\lfloor \log_2(m) - 1 \rfloor}$ possibilities
- Otherwise, play it safe and eliminate $\lfloor n/2 \rfloor$

Restricted Question Space

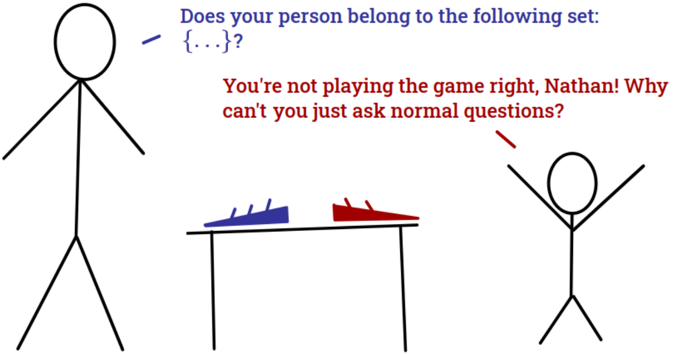


Figure: The truly visionary will never be without their critics.

RESTRICTED QUESTION SPACE

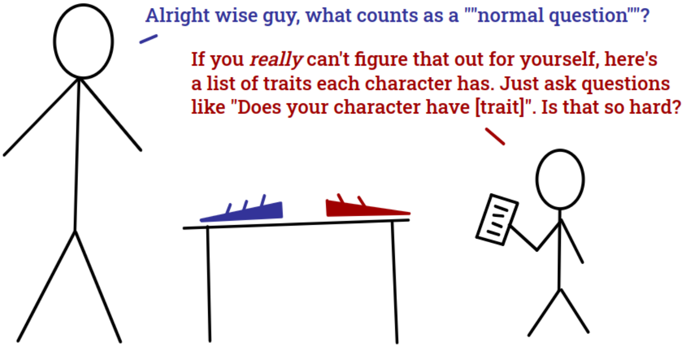


Figure: I'm not gonna let the man [my younger brother] tell me what to do [ask me to play this game in good faith]!

IS THAT SO HARD? YES.

Input:

- Set of $\mathcal{X}$ of items (characters)
- List of traits for each $x \in \mathcal{X}$
- Distribution X over $\mathcal{X}$
- Goal value k

Output:
Does there exist a guessing strategy, only making guesses of the form "does the item have trait T", with

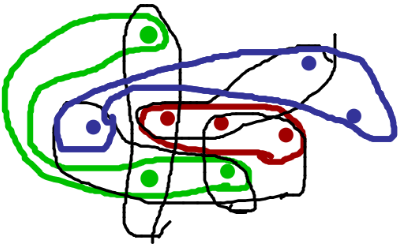
$$\mathbb{E}_{x \leftarrow X}[\text{\# of guesses until } x \text{ is uniquely identified}] \leq k?$$

IS THAT SO HARD? YES.

Known NP-complete problem: Exact Cover by 3-Sets (X3C)
Input:

- List of items in the universe
- Collection of 3-element sets of items

Output:
Does there exist a collection of those sets such that every item is contained in exactly one?



IS THAT SO HARD? YES.

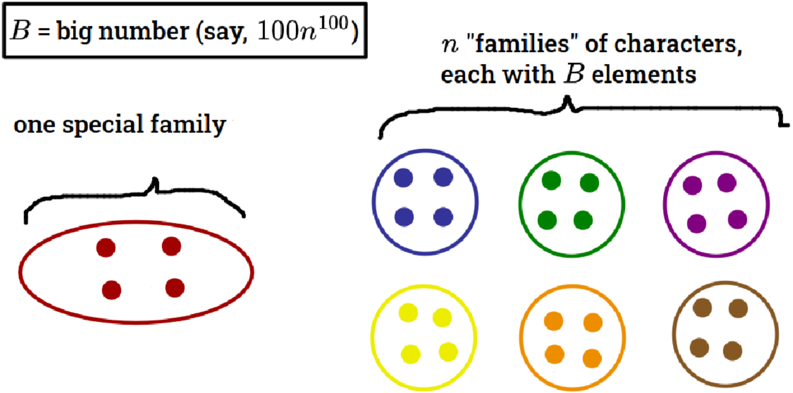


Figure: The characters involved in our reduction

IS THAT SO HARD? YES.

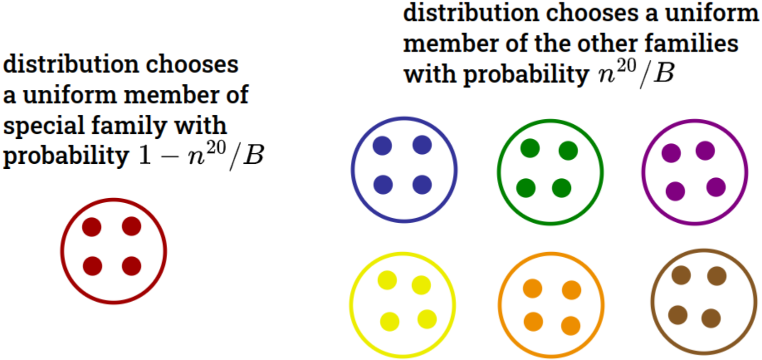
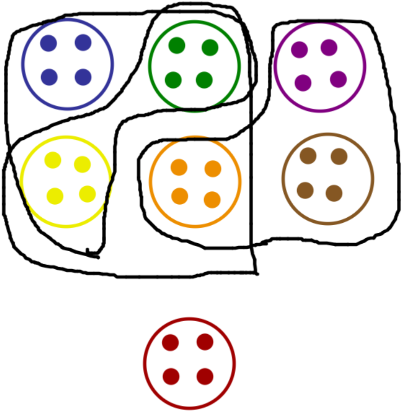


Figure: The distribution used in our reduction (could be modified to use uniform)

Is that so hard? Yes.



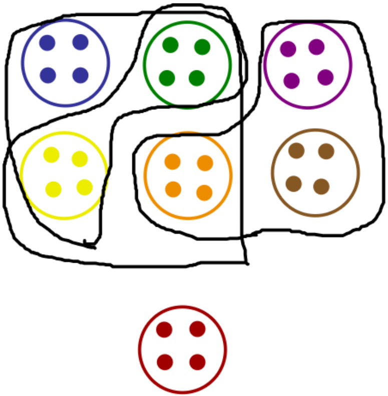
Traits:
- each character has a unique trait

- each family has a unique trait, except the special family

- each other trait is associated with 3 different families, encoding a chosen instance of X3C

Figure: The allowed traits in our reduction

Is that so hard? Yes.



Analysis:

- B is so large that it is never optimal to start checking individual characters until you've fully determined family

- Special family is almost always right, but can't rule out the others until a set has captured each of them

- If exists exact cover, can do in $n/3$ questions. Otherwise, need $n/3+1$.

- Set $k = B/2 + n/3$

Conclusion

Thank you to Max for mentoring us, and to DRP for support!