Class Road Map - IT-I

- L1 (9/26): Overview, Communications, Information, Entropy
- L2 (10/1): Props. Entropy, Mutual Information,
- L5 (10/10): AEP, Compression
- L6 (10/15): Compression, Method of Types,
- L7 (10/17): Types, U. Coding., Stoc. Processes, Entropy rates,
- L8 (10/22): Entropy rates, HMMs, Coding, Kraft,
- L9 (10/24): Kraft, Shannon Codes, Huffman, Shannon/Fano/Elias

- L10 (10/29): 
- L12
- L13
- L14
- L15
- L16
- L17
- L18
- L19

Finals Week: December 12th–16th.
Cumulative Outstanding Reading

- Read chapters 1 and 2 in our book (Cover & Thomas, “Information Theory”) (including Fano's inequality).
- Chapter 3 in our book (Cover & Thomas, “Information Theory”).
- Section 11.1 (method of types).
- Chapter 4 and 5 in our book (Cover & Thomas, “Information Theory”).
Homework

Homework 4 out on our web page (http://j.ee.washington.edu/~bilmes/classes/ee514a_fall_2013/), due next Tuesday, Oct 29th, at 11:45pm.
Office hours, every week, Tuesdays 4:30-5:30pm. Can also reach me at that time via a canvas conference.

Midterm on Thursday, 10/31 in class. Covers everything up to and including homework 4 (today’s cumulative reading). We’ll have a review on 10/29.

Next lecture will conflict with Stephen Boyd’s lecture (which is at 3:30-4:20pm in room EEB-105, see http://www.ee.washington.edu/news/2013/boyd_lytle_lecture.html). In order to see the lecture, 1/2 of Tuesday’s lecture will be youtube only, and we’ll meet in person only from 2:30-3:20 giving us enough time to walk down to EEB. I’m going also to try to get a different room for Tuesday (watch the announcements).
On Midterm

On Midterm

- **When:** Thursday (Oct 31st, 2013).
- **Length:** 1 hour 50 minutes, in class.

Closed book. OK to have one side of one 8.5 x 11 inch sheet of paper on which you can write anything you wish. Can be computer printed or hand written. Can be used for the final as well (so save the sheet).
On Midterm

- Length 1 hour 50 minutes length, in class.
- Closed book.
On Midterm

- Length 1 hour 50 minutes length, in class.
- Closed book.
- OK to have one side of one 8.5 \times 11 \text{ inch sheet of paper on which you can write anything you wish.}
On Midterm

- Length 1 hour 50 minutes length, in class.
- Closed book.
- OK to have one side of one 8.5 × 11 inch sheet of paper on which you can write anything you wish.
- Can be computer printed or hand written.
On Midterm

- Length 1 hour 50 minutes length, in class.
- Closed book.
- OK to have one side of one 8.5 × 11 inch sheet of paper on which you can write anything you wish.
- Can be computer printed or hand written.
- Can be used for the final as well (so save the sheet).
What is entropy of this random walk

- So, the entropy of the random walk is

\[ H(X) = \text{(overall edge uncertainty)} \]
\[ - \text{(overall node uncertainty in stationary condition)} \]

- Intuition: As node entropy decreases while keeping edge uncertainty constant, the network becomes more concentrated,
  - fewer nodes are hubs, and the hubs that remain are widely connected (since edge entropy is fixed).
  - In such case (few well connected hubs), it is likely one will land on such a hub (in a random walk) and then will be faced with a wide variety of choice as to where to go next ⇒ increase in overall uncertainty of the walk.
  - If node entropy goes up with edge entropy fixed, then many nodes are hubs all with relatively low connectivity, so hitting them doesn’t provide much choice ⇒ random walk entropy goes down.
Hidden Markov models (HMMs)

- HMM: class of distributions $p(X_{1:n}, Y_{1:n})$ over $2n$ random variables that factor in a particular way, $n$ is variable or unbounded as in a stream.
- Easiest way to depict all of the factorization properties is to use a graphical model, as in the below, where $n = 5$:

Let $Y_1, Y_2, \ldots, Y_n$ be a stationary Markov chain.

Let $X_{1:n}$ be a random function of this Markov chain. I.e.,

$$X_i = \begin{cases} 
\phi_1(Y_i) & \text{with probability } p_1(Y_i) \\
\phi_2(Y_i) & \text{with probability } p_2(Y_i) \\
\vdots & \\
\phi_m(Y_i) & \text{with probability } p_m(Y_i) 
\end{cases} = \phi_N(Y_i)(Y_i) \quad (9.13)$$

where $N(Y_i) \in \{1, 2, \ldots, m\}$ itself is a r.v., maybe dependent on $Y_i$. 
So summarizing, the bounds on the HMM information rates, we have

$$H(X_n|X_{n-1}, \ldots, X_1, Y_1) \leq H(X) \leq H(X_n|X_{n-1}, \ldots, X_1)$$  \hspace{1cm} (9.24)$$

And also, we have

$$\lim_{n \to \infty} H(X_n|X_{n-1}, \ldots, X_1, Y_1) = H(X) = \lim_{n \to \infty} H(X_n|X_{n-1}, \ldots, X_1)$$  \hspace{1cm} (9.25)$$
We want to develop practical coding algorithms that still approach, or achieve, the entropy limit.

They might use the distribution $p(x)$ which is either given or is estimated in some way.

We won’t get into any details on how to estimate $p(x)$ (that is a density estimation problem) but we assume we either have it or some approximation.

We will ultimately look, however, at what happens if the true distribution is $p(x)$ and we use $q(x)$ instead.
### Source Code

**Definition 9.2.2 (source code)**

A source code $C$ for r.v. $X$ is a mapping

$$C : \mathcal{X} \rightarrow \mathcal{D}^* = \{\mathcal{D} \cup (\mathcal{D} \times \mathcal{D}) \cup (\mathcal{D} \times \mathcal{D} \times \mathcal{D}) \cup \ldots\}$$  \hspace{1cm} (9.24)

from $\mathcal{X}$ to $\mathcal{D}^*$, the set of finite strings from a $D$-ary alphabet. $C(x)$ is the codeword corresponding to $x$, and $\ell(x)$ is the length of the codeword.

**Example 9.2.3**

Let $\mathcal{X} = \{\text{red}, \text{blue}\}$. Then a code might be $C(\text{red}) = 00$ and $C(\text{blue}) = 11$, which would be a binary code for $\mathcal{D} = \{0, 1\}$.

**Definition 9.2.4 (expected length)**

The expected length $L(C)$ of code $C$ for r.v. $X$ with distribution $p(x)$ is

$$L(C) = \sum_x p(x)\ell(x)$$  \hspace{1cm} (9.25)
Code types: non-singular

Definition 9.2.3 (non-singular)

A code is said to be non-singular if every element of the range of $X$ (i.e., all elements of $\mathcal{X}$) maps to a different string in $\mathcal{D}^*$. I.e.,

$$x_i \neq x_j \Rightarrow C(x_i) \neq C(x_j)$$

(9.24)

- We can view this as a mapping. It is less strict than onto but sufficient for being able to decode individual symbols.

- Note that $C_1$ above is singular.
Our goal, and definition of Code Extension

- Before going further, note: our goal is to send or store a sequence of code words for a sequence of symbols.
- A non-singular code could be unique if $\exists$ a comma between code words (e.g., Morse code is such that there is a space).
- In general, however, it is better to have a self punctuating or instantaneous code.

**Definition 9.2.3 (code extension)**

A code extension $C^*$ of $C$ is a mapping from finite length strings of $D$, defined as:

$$C^*(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2) \ldots C(x_n) \quad (9.24)$$

- Note that there are no commas in the extension, rather concatenation.
- Ex: If $C(x_1) = 0$ and $C(x_2) = 1$ then $C(x_1, x_2) = 01$. 
Code types: uniquely decodable

**Definition 9.2.3 (uniquely decodable)**

A code $C$ with extension $C^*$ is **uniquely decodable** if the extension $C^*$ is non-singular.

- $C_1$ singular. Extension of $C_{\|}$ singular so $C_{\|}$ not uniquely decodable.
- But how long must we wait until we know the source? In some even uniquely decodable cases, we might need to wait until the end.
- Ex: consider the code

<table>
<thead>
<tr>
<th>$x$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C(x)$</td>
<td>10</td>
<td>00</td>
<td>11</td>
<td>110</td>
</tr>
</tbody>
</table>

- Is this code uniquely decodable? Yes.
Prefix codes

Definition 9.2.3 (prefix code)
A code is called a prefix code or an instantaneous code if no codeword is a prefix of any other codeword.

- We know the end of a codeword because it can’t be a prefix of any other codeword.
- Code in previous page is not prefix free, 11 was a prefix of 110 so we couldn’t decide between 11 or 110 until we could count the number of zeros.
- A prefix code is self-punctuating (since there are implicit punctuation marks between codewords).
- Prefix code $\Rightarrow$ uniquely decodable. But (as we saw) uniquely decodable $\not\Rightarrow$ prefix code.
Goal is to find a code with the shortest possible expected length.

From the above code class, we might think that we want to use codes from the largest class possible (since we might think we’re more likely to get shorter codes).

We can do better than entropy with non-singular codes, but we want lossless encoding \( x = \text{ungzip}(\text{gzip}(x)) \).
Kraft inequality

Theorem 9.3.1 (Kraft inequality)

For any instantaneous code (prefix code) over alphabet of size $D$, the codeword lengths $\ell_1, \ell_2, \ldots, \ell_m$ must satisfy

$$\sum_{i} D^{-\ell_i} \leq 1 \quad (9.1)$$
Kraft inequality

Theorem 9.3.1 (Kraft inequality)

For any instantaneous code (prefix code) over alphabet of size $D$, the codeword lengths $\ell_1, \ell_2, \ldots, \ell_m$ must satisfy

$$\sum_i D^{-\ell_i} \leq 1 \quad (9.1)$$

Conversely, given a set of codeword lengths satisfying the above inequality, $\exists$ an instantaneous code with these word lengths.
Kraft inequality

**Theorem 9.3.1 (Kraft inequality)**

For any instantaneous code (prefix code) over alphabet of size $D$, the codeword lengths $\ell_1, \ell_2, \ldots, \ell_m$ must satisfy

$$\sum_i D^{-\ell_i} \leq 1 \quad (9.1)$$

Conversely, given a set of codeword lengths satisfying the above inequality, $\exists$ an instantaneous code with these word lengths.

- Note: converse says there exists a code with these lengths, not that all codes with these lengths will satisfy the inequality.
Kraft inequality

**Theorem 9.3.1 (Kraft inequality)**

For any instantaneous code (prefix code) over alphabet of size $D$, the codeword lengths $\ell_1, \ell_2, \ldots, \ell_m$ must satisfy

$$\sum_i D^{-\ell_i} \leq 1 \quad (9.1)$$

Conversely, given a set of codeword lengths satisfying the above inequality, $\exists$ an instantaneous code with these word lengths.

- Note: converse says there exists a code with these lengths, not that all codes with these lengths will satisfy the inequality.
- Key point: for $\ell_i$ satisfying Kraft, no further restriction imposed by also wanting a prefix code, so we might as well use a prefix code (assuming it is easy to find given the lengths)
**Theorem 9.3.1 (Kraft inequality)**

For any instantaneous code (prefix code) over alphabet of size $D$, the codeword lengths $\ell_1, \ell_2, \ldots, \ell_m$ must satisfy

$$\sum_i D^{-\ell_i} \leq 1 \quad (9.1)$$

Conversely, given a set of codeword lengths satisfying the above inequality, $\exists$ an instantaneous code with these word lengths.

- Note: converse says there exists a code with these lengths, not that all codes with these lengths will satisfy the inequality.
- Key point: for $\ell_i$ satisfying Kraft, no further restriction imposed by also wanting a prefix code, so we might as well use a prefix code (assuming it is easy to find given the lengths)
- Connects code existence to mathematical property on lengths!
  
Given Kraft lengths, can construct an instantaneous code (as we will see). Given lengths, can compute $E[\ell]$ and compare with $H$. 

proof of Kraft inequality.

- Represent set of codes on a $D$-ary (not necessarily balanced) tree:

  \[
  \begin{array}{cccc}
  & & 1 & \\
  & 2 & & \\
  D & & & \\
  \end{array}
  \]

  Codewords correspond to leaves
  Path from root to leaf determines a codeword
  Prefix condition: won't get to a codeword until we get to a leaf (no descendants of codewords are codewords)
proof of Kraft inequality.

- Represent set of codes on a $D$-ary (not necessarily balanced) tree:

  Codewords correspond to leaves.
  Path from root to leaf determines a codeword.
  Prefix condition: won’t get to a codeword until we get to a leaf (no descendants of codewords are codewords).
proof of Kraft inequality.

- Represent set of codes on a $D$-ary (not necessarily balanced) tree:
proof of Kraft inequality.

- Represent set of codes on a $D$-ary (not necessarily balanced) tree:
proof of Kraft inequality.

- Represent set of codes on a $D$-ary (not necessarily balanced) tree:

  ![Tree Diagram]

  - Codewords correspond to leaves

  Path from root to leaf determines a codeword.

  Prefix condition: won't get to a codeword until we get to a leaf (no descendants of codewords are codewords).
proof of Kraft inequality.

- Represent set of codes on a $D$-ary (not necessarily balanced) tree:

- Codewords correspond to leaves
- Path from root to leaf determines a codeword
proof of Kraft inequality.

- Represent set of codes on a $D$-ary (not necessarily balanced) tree:

  ![Tree Diagram]

  - Codewords correspond to leaves
  - Path from root to leaf determines a codeword
  - Prefix condition: won’t get to a codeword until we get to a leaf (no descendants of codewords are codewords)
Kraft inequality

... proof of Kraft inequality cont.

- $\ell_{\text{max}} = \max_i (\ell_i)$ is the length of the longest codeword.
proof of Kraft inequality cont.

- $\ell_{\text{max}} = \max_i(\ell_i)$ is the length of the longest codeword.
- We can expand the full-tree down to depth $\ell_{\text{max}}$. 
Kraft inequality

... proof of Kraft inequality cont.

- $\ell_{\text{max}} = \max_i (\ell_i)$ is the length of the longest codeword.
- We can expand the full-tree down to depth $\ell_{\text{max}}$.
Kraft inequality

\[ \ell_{\text{max}} = \max_i \ell_i \] is the length of the longest codeword.

We can expand the full-tree down to depth \( \ell_{\text{max}} \).

\[ \text{Some nodes at level } \ell_{\text{max}} \text{ are either:} \]

1. \( \text{codewords,} \)
2. \( \text{descendants of codewords,} \)
3. \( \text{neither} \)

Consider a codeword \( i \) at level \( \ell_i \) in tree (so it has length \( \ell_i \)).

Then, there are \( D \ell_{\text{max}} - \ell_i \) descendants in the tree at level \( \ell_{\text{max}} \).

Because of prefix condition, descendants of code \( i \) at level \( \ell_i \) are disjoint from descendants of code \( j \) at level \( \ell_j \) when \( i \neq j \) (i.e., descendant sets for different codewords are disjoint).

Also, total number of nodes in set of all descendants is \( \leq D \ell_{\text{max}} \).
Kraft inequality

... proof of Kraft inequality cont.

- $\ell_{\text{max}} = \max_i(\ell_i)$ is the length of the longest codeword.
- We can expand the full-tree down to depth $\ell_{\text{max}}$.

Some nodes at that level $\ell_{\text{max}}$ are either:

- 1 codewords,
- 2 descendants of codewords, or
- 3 neither.

Consider a codeword $i$ at level $\ell_i$ in tree (so it has length $\ell_i$).
Then, there are $D_{\ell_{\text{max}} - \ell_i}$ descendants in the tree at level $\ell_{\text{max}}$.

Because of prefix condition, descendants of code $i$ at level $\ell_i$ are disjoint from descendants of code $j$ at level $\ell_j$ when $i \neq j$ (i.e., descendant sets for different codewords are disjoint).

Also, total number of nodes in set of all descendants is $\leq D_{\ell_{\text{max}}}$.
Kraft inequality

... proof of Kraft inequality cont.

- $\ell_{\text{max}} = \max_i (\ell_i)$ is the length of the longest codeword.
- We can expand the full-tree down to depth $\ell_{\text{max}}$. Some nodes at that level $\ell_{\text{max}}$ are either:
  1. codewords,
Kraft inequality

...proof of Kraft inequality cont.

- \( \ell_{\text{max}} = \max_i (\ell_i) \) is the length of the longest codeword.
- We can expand the full-tree down to depth \( \ell_{\text{max}} \).

Some nodes at that level \( \ell_{\text{max}} \) are either:

1. codewords,
2. descendants of codewords, or
\[ \ell_{\text{max}} = \max_i (\ell_i) \] is the length of the longest codeword.

We can expand the full-tree down to depth \( \ell_{\text{max}} \).

Some nodes at that level \( \ell_{\text{max}} \) are either:

1. codewords,
2. descendants of codewords, or
3. neither
Kraft inequality

... proof of Kraft inequality cont.

- \( \ell_{\text{max}} = \max_i (\ell_i) \) is the length of the longest codeword.
- We can expand the full-tree down to depth \( \ell_{\text{max}} \).
  
  Some nodes at that level \( \ell_{\text{max}} \) are either:
  
  1. codewords,
  2. descendants of codewords, or
  3. neither

- Consider a codeword \( i \) at level \( \ell_i \) in tree (so it has length \( \ell_i \)).
Kraft inequality

... proof of Kraft inequality cont.

- \( \ell_{\text{max}} = \max_i (\ell_i) \) is the length of the longest codeword.
- We can expand the full-tree down to depth \( \ell_{\text{max}} \).

Some nodes at that level \( \ell_{\text{max}} \) are either:

1. codewords,
2. descendants of codewords, or
3. neither

Consider a codeword \( i \) at level \( \ell_i \) in tree (so it has length \( \ell_i \)).

Then, there are \( D^{\ell_{\text{max}} - \ell_i} \) descendants in the tree at level \( \ell_{\text{max}} \).
... proof of Kraft inequality cont.

- \( \ell_{\max} = \max_i (\ell_i) \) is the length of the longest codeword.
- We can expand the full-tree down to depth \( \ell_{\max} \).
  
  Some nodes at that level \( \ell_{\max} \) are either:
  1. codewords,
  2. descendants of codewords, or
  3. neither

- Consider a codeword \( i \) at level \( \ell_i \) in tree (so it has length \( \ell_i \)).
- Then, there are \( D^{\ell_{\max} - \ell_i} \) descendants in the tree at level \( \ell_{\max} \).
- Because of prefix condition, descendants of code \( i \) at level \( \ell_i \) are disjoint from descendants of code \( j \) at level \( \ell_j \) when \( i \neq j \) (i.e., descendant sets for different codewords are disjoint).
Kraft inequality

... proof of Kraft inequality cont.

- $\ell_{\text{max}} = \max_i (\ell_i)$ is the length of the longest codeword.
- We can expand the full-tree down to depth $\ell_{\text{max}}$.
  Some nodes at that level $\ell_{\text{max}}$ are either:
  1. codewords,
  2. descendants of codewords, or
  3. neither

- Consider a codeword $i$ at level $\ell_i$ in tree (so it has length $\ell_i$).
- Then, there are $D^{\ell_{\text{max}} - \ell_i}$ descendants in the tree at level $\ell_{\text{max}}$.
- Because of prefix condition, descendants of code $i$ at level $\ell_i$ are disjoint from descendants of code $j$ at level $\ell_j$ when $i \neq j$ (i.e., descendant sets for different codewords are disjoint).
- Also, total number of nodes in set of all descendants is $\leq D^{\ell_{\text{max}}}$. ...
Kraft inequality

... proof of Kraft inequality cont.

- All of the above implies:

\[ \sum_i D^{\ell_{\text{max}} - \ell_i} \leq D^{\ell_{\text{max}}} \Rightarrow \sum_i D^{-\ell_i} \leq 1 \quad (9.2) \]
... proof of Kraft inequality cont.

- All of the above implies:

\[ \sum_{i} D^{\ell_{\max} - \ell_i} \leq D^{\ell_{\max}} \Rightarrow \sum_{i} D^{-\ell_i} \leq 1 \] \hspace{1cm} (9.2)

- Conversely: given codeword lengths \( \ell_1, \ell_2, \ldots, \ell_m \) satisfying Kraft inequality (we must construct a prefix code with these lengths).
Kraft inequality

... proof of Kraft inequality cont.

- All of the above implies:

\[ \sum_{i} D^{\ell_{\text{max}} - \ell_i} \leq D^{\ell_{\text{max}}} \Rightarrow \sum_{i} D^{-\ell_i} \leq 1 \quad (9.2) \]

- Conversely: given codeword lengths \( \ell_1, \ell_2, \ldots, \ell_m \) satisfying Kraft inequality (we must construct a prefix code with these lengths).

- Consider a full \( D \)-ary tree of depth \( \ell_{\text{max}} \) with \( D^{\ell_{\text{max}}} \) terminal nodes.
Kraft inequality

... proof of Kraft inequality cont.

- All of the above implies:

\[ \sum_i D^{\ell_{\text{max}} - \ell_i} \leq D^{\ell_{\text{max}}} \Rightarrow \sum_i D^{-\ell_i} \leq 1 \quad (9.2) \]

- Conversely: given codeword lengths \( \ell_1, \ell_2, \ldots, \ell_m \) satisfying Kraft inequality (we must construct a prefix code with these lengths).

- Consider a full \( D \)-ary tree of depth \( \ell_{\text{max}} \) with \( D^{\ell_{\text{max}}} \) terminal nodes.

  - @ level 0, \( \exists \) fraction 1 of the descendants at each node at that level;
  - @ level 1, \( \exists \) fraction \( 1/D \) descendants at each node at that level;
  - @ level 2, \( \exists \) fraction \( 1/D^2 \) . . .
All of the above implies:

$$\sum_i D^{\ell_{\text{max}} - \ell_i} \leq D^{\ell_{\text{max}}} \quad \Rightarrow \quad \sum_i D^{-\ell_i} \leq 1 \quad (9.2)$$

Conversely: given codeword lengths $\ell_1, \ell_2, \ldots, \ell_m$ satisfying Kraft inequality (we must construct a prefix code with these lengths).

Consider a full $D$-ary tree of depth $\ell_{\text{max}}$ with $D^{\ell_{\text{max}}}$ terminal nodes.

@ level 0, $\exists$ fraction 1 of the descendants at each node at that level;
@ level 1, $\exists$ fraction $1/D$ descendants at each node at that level;
@ level 2, $\exists$ fraction $1/D^2$ . . .

In general, at each level $i \in [0, \ell_{\text{max}}]$ in tree, there is a fraction $D^{-i}$ terminal nodes that are descendants that stem from each of the $D^i$ nodes at level $i$. . .
proposition of Kraft inequality cont.

- Sort the lengths \((\ell_1, \ell_2, \ldots, \ell_m)\) ascending to \((s_1, s_2, \ldots, s_m)\) with \(s_1 \leq s_2 \leq \cdots \leq s_m\). Note there are as many lengths as there are codewords.
Kraft inequality

... proof of Kraft inequality cont.

- Sort the lengths \((\ell_1, \ell_2, \ldots, \ell_m)\) ascending to \((s_1, s_2, \ldots, s_m)\) with \(s_1 \leq s_2 \leq \cdots \leq s_m\). Note there are as many lengths as there are codewords.

- For length \(s_1\) chose any node at level \(s_1\) to indicate the code.
Kraft inequality

... proof of Kraft inequality cont.

- Sort the lengths \((\ell_1, \ell_2, \ldots, \ell_m)\) ascending to \((s_1, s_2, \ldots, s_m)\) with \(s_1 \leq s_2 \leq \cdots \leq s_m\). Note there are as many lengths as there are codewords.

- For length \(s_1\) chose any node at level \(s_1\) to indicate the code.

- To ensure prefix free property, the node becomes a terminal node, thus eliminating a fraction \(D^{-s_1}\) of the terminal nodes at depth \(\ell_{\text{max}}\) (which would have been potential code words of longer length, but now they are out of the running).
Sort the lengths \((\ell_1, \ell_2, \ldots, \ell_m)\) ascending to \((s_1, s_2, \ldots, s_m)\) with \(s_1 \leq s_2 \leq \cdots \leq s_m\). Note there are as many lengths as there are codewords.

For length \(s_1\) chose any node at level \(s_1\) to indicate the code.

To ensure prefix free property, the node becomes a terminal node, thus eliminating a fraction \(D^{-s_1}\) of the terminal nodes at depth \(\ell_{\text{max}}\) (which would have been potential code words of longer length, but now they are out of the running).

Next: chose any remaining node at level \(s_2\) (we have \((D^{s_1} - 1)D^{s_2-s_1} > 0\) choices at this point), thus eliminating a fraction \(D^{-s_2}\) of the nodes.

...
Kraft inequality

...proof of Kraft inequality cont.

- Sort the lengths \((\ell_1, \ell_2, \ldots, \ell_m)\) ascending to \((s_1, s_2, \ldots, s_m)\) with \(s_1 \leq s_2 \leq \cdots \leq s_m\). Note there are as many lengths as there are codewords.

- For length \(s_1\) chose any node at level \(s_1\) to indicate the code.

- To ensure prefix free property, the node becomes a terminal node, thus eliminating a fraction \(D^{-s_1}\) of the terminal nodes at depth \(\ell_{\text{max}}\) (which would have been potential code words of longer length, but now they are out of the running).

- Next: chose any remaining node at level \(s_2\) (we have \((D^{s_1} - 1)D^{s_2-s_1} > 0\) choices at this point), thus eliminating a fraction \(D^{-s_2}\) of the nodes.

- Total fraction eliminated is \(D^{-s_1} + D^{-s_2}\).
Kraft inequality

... proof of Kraft inequality cont.

- Continuing this process, we eliminate a fraction $\sum_{i=1}^{m} D^{-s_i}$ of the nodes, while retaining that the code is instantaneous (a codeword can’t be a prefix of another).
Kraft inequality

\[ \sum_{i=1}^{m} D^{-s_i} \]

Continuing this process, we eliminate a fraction \[ \sum_{i=1}^{m} D^{-s_i} \] of the nodes, while retaining that the code is instantaneous (a codeword can’t be a prefix of another).

But since by assumption \[ \sum_{i=1}^{m} D^{-s_i} \leq 1 \] we never eliminate more than all of the codewords, so this process won’t run out of codewords.
Kraft inequality

...proof of Kraft inequality cont.

- Continuing this process, we eliminate a fraction \( \sum_{i=1}^{m} D^{-s_i} \) of the nodes, while retaining that the code is instantaneous (a codeword can’t be a prefix of another).

- But since by assumption \( \sum_{i=1}^{m} D^{-s_i} \leq 1 \) we never eliminate more than all of the codewords, so this process won’t run out of codewords.

- Thus, we have created a prefix-free code with the desired lengths.
Infinite Kraft

Theorem 9.3.2 (countably infinite Kraft)

For any countably infinite set of codewords that form a prefix set, this satisfies the extended Kraft inequality, i.e.

$$\sum_{i=1}^{\infty} D^{-\ell_i} \leq 1 \quad (9.3)$$

Conversely, given $\ell_i$ satisfying the above, $\exists$ a prefix code with these lengths.

proof of countably infinite Kraft.

- Assume we have such a prefix code, and let the $D$-ary alphabet be $\{0, 1, \ldots, D - 1\}$. 

...
Theorem 9.3.2 (countably infinite Kraft)

For any countably infinite set of codewords that form a prefix set, this satisfies the extended Kraft inequality, i.e.

\[
\sum_{i=1}^{\infty} D^{-\ell_i} \leq 1 \tag{9.3}
\]

Conversely, given \( \ell_i \) satisfying the above, \( \exists \) a prefix code with these lengths.

proof of countably infinite Kraft.

- Assume we have such a prefix code, and let the \( D \)-ary alphabet be \( \{0, 1, \ldots, D - 1\} \).
- Consider the \( i^{th} \) codeword \( y_1, y_2, \ldots, y_{\ell_i} \).
Kraft inequality

...proof of infinite Kraft.

- Consider expansion of codeword using fractional digits:

\[ 0.y_1y_2y_3 \ldots y_{\ell_i} = \sum_{j=1}^{\ell_i} y_j D^{-j} \]  \hspace{1cm} (9.4)
Kraft inequality

... proof of infinite Kraft.

- Consider expansion of codeword using fractional digits:

\[ 0.y_1y_2y_3\ldots y_{\ell_i} = \sum_{j=1}^{\ell_i} y_j D^{-j} \]  
(9.4)

- Examples: When \( D = \{0, 1\} \) then 0.1 = 1/2, 0.01 = 1/4, 0.11 = 3/4, and 0.001 = 1/8 (so bits are after the binary point). 

Kraft inequality

... proof of infinite Kraft.

- Consider expansion of codeword using fractional digits:
  \[ 0.y_1y_2y_3 \ldots y_{\ell_i} = \sum_{j=1}^{\ell_i} y_j D^{-j} \quad (9.4) \]

- Examples: When \( D = \{0, 1\} \) then 0.1 = 1/2, 0.01 = 1/4, 0.11 = 3/4, and 0.001 = 1/8 (so bits are after the binary point).

- Associate each codeword \( y_1:_{\ell_i} \) with the half-open interval on the real line \([0.y_1y_2 \ldots y_{\ell_i} , 0.y_1y_2 \ldots y_{\ell_i} + 1/D^{\ell_i})\)
Kraft inequality

... proof of infinite Kraft.

- Consider expansion of codeword using fractional digits:

\[ 0.y_1y_2y_3 \ldots y_{\ell_i} = \sum_{j=1}^{\ell_i} y_j D^{-j} \quad (9.4) \]

- Examples: When \( D = \{0, 1\} \) then \( 0.1 = 1/2 \), \( 0.01 = 1/4 \), \( 0.11 = 3/4 \), and \( 0.001 = 1/8 \) (so bits are after the binary point).

- Associate each codeword \( y_{1:\ell_i} \) with the half-open interval on the real line \([0.y_1y_2 \ldots y_{\ell_i}, 0.y_1y_2 \ldots y_{\ell_i} + 1/D^{\ell_i}]\)  

- Example: With \( D = 10 \), then if \( 0.y_1y_2y_3 = 0.157 \), the associated half-open interval is \([0.157, 0.158)\), and if \( 0.y_1y_2y_3 = 0.159 \), the associated half-open interval is \([0.159, 0.160)\)
Consider expansion of codeword using fractional digits:

\[ 0.y_1y_2y_3 \ldots y_{\ell_i} = \sum_{j=1}^{\ell_i} y_j D^{-j} \]  

Examples: When \( D = \{0, 1\} \) then \( 0.1 = 1/2, 0.01 = 1/4, 0.11 = 3/4, \) and \( 0.001 = 1/8 \) (so bits are after the binary point).

Associate each codeword \( y_1:\ell_i \) with the half-open interval on the real line \([0.y_1y_2 \ldots y_{\ell_i}, 0.y_1y_2 \ldots y_{\ell_i} + 1/D^{\ell_i}]\)

Example: With \( D = 10 \), then if \( 0.y_1y_2y_3 = 0.157 \), the associated half-open interval is \([0.157, 0.158)\), and if \( 0.y_1y_2y_3 = 0.159 \), the associated half-open interval is \([0.159, 0.160)\).
Kraft inequality

...proof of infinite Kraft.

- So the interval for codeword $y_1y_2y_3\ldots y_{\ell_i}$ corresponds to the set of all real numbers that begins with $0.y_1y_2y_3\ldots y_{\ell_i}$.
Kraft inequality

...proof of infinite Kraft.

So the interval for codeword $y_1y_2y_3 \ldots y_{\ell_i}$ corresponds to the set of all real numbers that begins with $0.y_1y_2y_3 \ldots y_{\ell_i}$ and is thus a sub-interval of the unit interval.
Kraft inequality

... proof of infinite Kraft.

- So the interval for codeword $y_1y_2y_3 \ldots y_{\ell_i}$ corresponds to the set of all real numbers that begins with $0.y_1y_2y_3 \ldots y_{\ell_i}$ and is thus a sub-interval of the unit interval.

- Also $y_1y_2y_3 \ldots y_{\ell_i}$ is not a prefix of any other codeword, so the intervals must be disjoint.
Kraft inequality

... proof of infinite Kraft.

- So the interval for codeword $y_1y_2y_3\ldots y_{\ell_i}$ corresponds to the set of all real numbers that begins with $0.y_1y_2y_3\ldots y_{\ell_i}$ and is thus a sub-interval of the unit interval.

- Also $y_1y_2y_3\ldots y_{\ell_i}$ is not a prefix of any other codeword, so the intervals must be disjoint.

- Length of interval for codeword $y_1y_2y_3\ldots y_{\ell_i}$ is $D^{-\ell_i}$. 

And since all intervals live in $[0,1)$ we must have 

$$\sum_{i} D^{-\ell_i} \leq 1 \quad \text{(9.5)}$$

Proof of converse is similar to finite case and also to arithmetic coding that we'll soon see, so we skip the proof here.
Kraft inequality

... proof of infinite Kraft.

- So the interval for codeword $y_1 y_2 y_3 \ldots y_{\ell_i}$ corresponds to the set of all real numbers that begins with $0.y_1 y_2 y_3 \ldots y_{\ell_i}$ and is thus a sub-interval of the unit interval.

- Also $y_1 y_2 y_3 \ldots y_{\ell_i}$ is not a prefix of any other codeword, so the intervals must be disjoint.

- Length of interval for codeword $y_1 y_2 y_3 \ldots y_{\ell_i}$ is $D^{-\ell_i}$.

- And since all intervals live in $[0, 1)$ we must have

$$\sum_i D^{-\ell_i} \leq 1$$

(9.5)
Kraft inequality

...proof of infinite Kraft.

- So the interval for codeword $y_1y_2y_3 \ldots y_{\ell_i}$ corresponds to the set of all real numbers that begins with $0.y_1y_2y_3 \ldots y_{\ell_i}$ and is thus a sub-interval of the unit interval.

- Also $y_1y_2y_3 \ldots y_{\ell_i}$ is not a prefix of any other codeword, so the intervals must be disjoint.

- Length of interval for codeword $y_1y_2y_3 \ldots y_{\ell_i}$ is $D^{-\ell_i}$.

- And since all intervals live in $[0, 1)$ we must have

$$\sum_i D^{-\ell_i} \leq 1 \quad (9.5)$$

- Proof of converse is similar to finite case and also to arithmetic coding that we’ll soon see, so we skip the proof here.
Towards Optimal Codes

- Summarizing: Prefix code $\Leftrightarrow$ Kraft inequality.
Towards Optimal Codes

- Summarizing: Prefix code $\Leftrightarrow$ Kraft inequality.
- Thus, we need only find lengths that satisfy Kraft to find a prefix code.
Towards Optimal Codes

- Summarizing: Prefix code ⇔ Kraft inequality.
- Thus, we need only find lengths that satisfy Kraft to find a prefix code.
- Goal: find a prefix code with minimum expected length

\[ L(C) = \sum_i p_i \ell_i \quad (9.6) \]

This is a constrained optimization problem:

\[ \min \{ \ell_1 : m \in \mathbb{Z}^m_+ \} \]

subject to

\[ \sum_i D - \ell_i \leq 1 \]

Integer program is an NP-complete optimization, not likely to be efficiently solvable (unless P=NP).
Towards Optimal Codes

- Summarizing: Prefix code $\Leftrightarrow$ Kraft inequality.
- Thus, we need only find lengths that satisfy Kraft to find a prefix code.
- Goal: find a prefix code with minimum expected length
  \[ L(C) = \sum_i p_i \ell_i \]  
  \[ (9.6) \]
- This is a constrained optimization problem:
  \[
  \min_{\{\ell_1, \ldots, \ell_m\} \in \mathbb{Z}_+^m} \sum_i p_i \ell_i \\
  \text{subject to} \quad \sum_i D^{-\ell_i} \leq 1 
  \]  
  \[ (9.7) \]
Towards Optimal Codes

- Summarizing: Prefix code ⇔ Kraft inequality.
- Thus, we need only find lengths that satisfy Kraft to find a prefix code.
- Goal: find a prefix code with minimum expected length

\[ L(C) = \sum_i p_i \ell_i \]  \hspace{1cm} (9.6)

- This is a constrained optimization problem:

\[ \min_{\{\ell_1: m\} \in \mathbb{Z}^m_+} \sum_i p_i \ell_i \]  \hspace{1cm} (9.7)

subject to \[ \sum_i D^{-\ell_i} \leq 1 \]

- Integer program is an NP-complete optimization, not likely to be efficiently solvable (unless P=NP).
Towards Optimal Codes

- Relax the integer constraints on $\ell_i$ for now, and consider Lagrangian

$$J = \sum_i p_i \ell_i + \lambda \left( \sum_i D^{-\ell_i} - 1 \right)$$  \hspace{1cm} (9.8)
Towards Optimal Codes

- Relax the integer constraints on $\ell_i$ for now, and consider Lagrangian

$$J = \sum_i p_i \ell_i + \lambda \left( \sum_i D^{-\ell_i} - 1 \right)$$  \hspace{1cm} (9.8)

- Taking derivatives and setting to 0,

$$\frac{\partial J}{\partial \ell_i}$$  \hspace{1cm} (9.10)

$$D^{-\ell_i} = p_i \lambda \ln D$$  \hspace{1cm} (9.11)

$$\ell_i^* = -\log D p_i$$  \hspace{1cm} (9.12)
Towards Optimal Codes

- Relax the integer constraints on $\ell_i$ for now, and consider Lagrangian

$$J = \sum_i p_i \ell_i + \lambda \left( \sum_i D^{-\ell_i} - 1 \right) \quad (9.8)$$

- Taking derivatives and setting to 0,

$$\frac{\partial J}{\partial \ell_i} = p_i - \lambda D^{-\ell_i} \ln D = 0 \quad (9.9)$$

$$\Rightarrow D^{-\ell_i} = \frac{p_i}{\lambda \ln D} \quad (9.10)$$

$$\Rightarrow \ell_i^* = - \log_D p_i \quad (9.12)$$
Towards Optimal Codes

- Relax the integer constraints on $\ell_i$ for now, and consider Lagrangian

\[
J = \sum_i p_i \ell_i + \lambda \left( \sum_i D^{-\ell_i} - 1 \right)
\]  \hspace{1cm} (9.8)

- Taking derivatives and setting to 0,

\[
\frac{\partial J}{\partial \ell_i} = p_i - \lambda D^{-\ell_i} \ln D = 0
\]  \hspace{1cm} (9.9)

\[
\Rightarrow D^{-\ell_i} = \frac{p_i}{\lambda \ln D}
\]  \hspace{1cm} (9.10)

\[
\Rightarrow \ell_i^* = -\log_D p_i
\]  \hspace{1cm} (9.12)
Towards Optimal Codes

- Relax the integer constraints on $\ell_i$ for now, and consider Lagrangian

$$J = \sum_i p_i \ell_i + \lambda \left( \sum_i D^{-\ell_i} - 1 \right)$$  \hspace{1cm} (9.8)

- Taking derivatives and setting to 0,

$$\frac{\partial J}{\partial \ell_i} = p_i - \lambda D^{-\ell_i} \ln D = 0$$  \hspace{1cm} (9.9)

$$\Rightarrow D^{-\ell_i} = \frac{p_i}{\lambda \ln D}$$  \hspace{1cm} (9.10)

$$\frac{\partial J}{\partial \lambda}$$

$$\Rightarrow$$
Towards Optimal Codes

- Relax the integer constraints on $\ell_i$ for now, and consider Lagrangian

$$J = \sum_i p_i \ell_i + \lambda (\sum_i D^{-\ell_i} - 1)$$  \quad (9.8)

- Taking derivatives and setting to 0,

$$\frac{\partial J}{\partial \ell_i} = p_i - \lambda D^{-\ell_i} \ln D = 0$$  \quad (9.9)

$$\Rightarrow D^{-\ell_i} = \frac{p_i}{\lambda \ln D}$$  \quad (9.10)

$$\frac{\partial J}{\partial \lambda} = \sum_i D^{-\ell_i} - 1$$

$$\Rightarrow$$

$$D^{-\ell_i} = \frac{p_i}{\lambda \ln D}$$

(9.12)
Towards Optimal Codes

- Relax the integer constraints on $\ell_i$ for now, and consider Lagrangian

$$J = \sum_i p_i \ell_i + \lambda \left( \sum_i D^{-\ell_i} - 1 \right)$$  \hspace{1cm} (9.8)

- Taking derivatives and setting to 0,

$$\frac{\partial J}{\partial \ell_i} = p_i - \lambda D^{-\ell_i} \ln D = 0$$ \hspace{1cm} (9.9)

$$\Rightarrow D^{-\ell_i} = \frac{p_i}{\lambda \ln D}$$ \hspace{1cm} (9.10)

$$\frac{\partial J}{\partial \lambda} = \sum_i D^{-\ell_i} - 1 = 0$$ \hspace{1cm} (9.12)
Towards Optimal Codes

- Relax the integer constraints on $\ell_i$ for now, and consider Lagrangian

$$J = \sum_i p_i \ell_i + \lambda \left( \sum_i D^{-\ell_i} - 1 \right)$$  \hspace{1cm} (9.8)

- Taking derivatives and setting to 0,

$$\frac{\partial J}{\partial \ell_i} = p_i - \lambda D^{-\ell_i} \ln D = 0$$ \hspace{1cm} (9.9)

$$\Rightarrow D^{-\ell_i} = \frac{p_i}{\lambda \ln D}$$ \hspace{1cm} (9.10)

$$\frac{\partial J}{\partial \lambda} = \sum_i D^{-\ell_i} - 1 = 0 \quad \Rightarrow \quad \lambda = 1/\ln D$$ \hspace{1cm} (9.11)
Towards Optimal Codes

- Relax the integer constraints on $\ell_i$ for now, and consider Lagrangian

$$J = \sum_i p_i \ell_i + \lambda \left( \sum_i D^{-\ell_i} - 1 \right)$$  \hspace{1cm} (9.8)

- Taking derivatives and setting to 0,

$$\frac{\partial J}{\partial \ell_i} = p_i - \lambda D^{-\ell_i} \ln D = 0 \Rightarrow D^{-\ell_i} = \frac{p_i}{\lambda \ln D}$$  \hspace{1cm} (9.9)

$$\frac{\partial J}{\partial \lambda} = \sum_i D^{-\ell_i} - 1 = 0 \Rightarrow \lambda = 1/\ln D$$  \hspace{1cm} (9.11)

$$\Rightarrow D^{-\ell_i} = p_i$$  \hspace{1cm} (9.12)
Towards Optimal Codes

- Relax the integer constraints on $\ell_i$ for now, and consider Lagrangian

$$J = \sum_i p_i \ell_i + \lambda \left( \sum_i D^{-\ell_i} - 1 \right) \quad (9.8)$$

- Taking derivatives and setting to 0,

$$\frac{\partial J}{\partial \ell_i} = p_i - \lambda D^{-\ell_i} \ln D = 0 \quad (9.9)$$

$$\Rightarrow D^{-\ell_i} = \frac{p_i}{\lambda \ln D} \quad (9.10)$$

$$\frac{\partial J}{\partial \lambda} = \sum_i D^{-\ell_i} - 1 = 0 \quad \Rightarrow \quad \lambda = 1/\ln D \quad (9.11)$$

$$\Rightarrow D^{-\ell_i} = p_i \quad \text{yielding} \quad \ell_i^* = -\log_D p_i \quad (9.12)$$
Towards Optimal Codes

- This implies that:

\[ L^* = \sum p_i \ell_i^* = \sum p_i \log_2 \frac{1}{p_i} = H(X) / \log_2 D \]

(9.13)

So the optimal expected code length, as a result of this optimization process, is the entropy assuming that we are allowed to have fractional code lengths.

Since \( \ell_i^* = -\log_2 p_i \), this means that optimal code “length” (while fractional) is the same as the information about the event. I.e., shortest possible coding length is the inherent information about an event. This is like the MDL (minimum description principle), tries to find the simplest explanation about a source.

Compare fractional codeword lengths to long block codes, what is the relation?
This implies that:

$$L^* = \sum_i p_i \ell_i^*$$

(9.13)
Towards Optimal Codes

- This implies that:

\[ L^* = \sum_i p_i \ell_i^* = - \sum_i p_i \log_D p_i \]  

(9.13)
Towards Optimal Codes

This implies that:

\[ L^* = \sum_i p_i \ell_i^* = - \sum_i p_i \log_D p_i = H_D(X) \]  

(9.13)
Towards Optimal Codes

This implies that:

\[ L^* = \sum_i p_i \ell_i^* = - \sum_i p_i \log_D p_i = H_D(X) = H(X) / \log D \] (9.13)

So the optimal expected code length, as a result of this optimization process, is the entropy assuming that we are allowed to have fractional code lengths. Since \( \ell_i^* = -\log_D p_i \), this means that optimal code “length” (while fractional) is the same as the information about the event. I.e., shortest possible coding length is the inherent information about an event. This is like the MDL (minimum description principle), tries to find the simplest explanation about a source.

Compare fractional codeword lengths to long block codes, what is the relation?
Towards Optimal Codes

- This implies that:

\[ L^* = \sum_i p_i \ell_i^* = - \sum_i p_i \log_D p_i = H_D(X) = H(X)/\log D \quad (9.13) \]

- So the optimal expected code length, as a result of this optimization process, is the entropy.
Towards Optimal Codes

This implies that:

\[ L^* = \sum_i p_i \ell_i^* = - \sum_i p_i \log D p_i = H_D(X) = H(X) / \log D \]  (9.13)

So the optimal expected code length, as a result of this optimization process, is the entropy assuming that we are allowed to have fractional code lengths.
Towards Optimal Codes

This implies that:

\[ L^* = \sum p_i \ell_i^* = -\sum p_i \log_D p_i = H_D(X) = H(X)/\log D \]  \hspace{1cm} (9.13)

So the optimal expected code length, as a result of this optimization process, is the entropy assuming that we are allowed to have fractional code lengths.

Since \( \ell_i^* = -\log_D p_i \), this means that optimal code “length” (while fractional) is the same as the information about the event. I.e., shortest possible coding length is the inherent information about an event. This is like the MDL (minimum description principle), tries to find the simplest explanation about a source.
Towards Optimal Codes

- This implies that:

\[ L^* = \sum_i p_i \ell_i^* = - \sum_i p_i \log D p_i = H_D(X) = H(X)/\log D \quad (9.13) \]

- So the optimal expected code length, as a result of this optimization process, is the entropy assuming that we are allowed to have fractional code lengths.

- Since \( \ell_i^* = - \log D p_i \), this means that optimal code “length” (while fractional) is the same as the information about the event. I.e., shortest possible coding length is the inherent information about an event. This is like the MDL (minimum description principle), tries to find the simplest explanation about a source.

- Compare fractional codeword lengths to long block codes, what is the relation?
Theorem 9.3.3

Entropy is the minimum expected length. That is, the expected length $L$ of any instantaneous $D$-ary code (which thus satisfies Kraft inequality) for a r.v. $X$ is such that

$$L \geq H_D(X)$$  \hspace{1cm} (9.14)

with equality iff $D^{-\ell_i} = p_i$. 


Proof of Theorem 9.3.3.

\[
L - H_D(X)
\]  

(9.15)
Proof of Theorem 9.3.3.

\[ L - H_D(X) = \sum_i p_i \ell_i - \sum_i p_i \log_D 1/p_i \]  
(9.15)

\[ = -\sum_i p_i \log_2 D^{-\ell_i} + \sum_i p_i \log_2 D \]  
(9.16)

\[ = -\sum_i p_i \log_2 D^{-\ell_i} + \log_2 D (\sum_i D^{-\ell_i}) - \log_2 D (\sum_i D^{-\ell_i}) \]  
(9.17)

\[ = \sum_i p_i \log_2 p_i r_i - \log_2 D (\sum_i D^{-\ell_i}) \]  
(9.18)

\[ \geq 0 \text{ since } c \leq 1 \text{ by Kraft, where } c = \sum_i D^{-\ell_i} \]  
(9.20)

...
Proof of Theorem 9.3.3.

\[ L - H_D(X) = \sum_i p_i \ell_i - \sum_i p_i \log_D 1/p_i \]  \hspace{1cm} (9.15)

\[ = - \sum_i p_i \log_D D^{-\ell_i} + \sum_i p_i \log_D p_i \]  \hspace{1cm} (9.16)

\[ = \sum_i p_i \log_D p_i - \log_D (\sum_i D^{-\ell_i}) \]  \hspace{1cm} (9.17)

\[ \geq 0 \text{ since } c \leq 1 \text{ by Kraft, where } c = \sum_i D^{-\ell_i} \]  \hspace{1cm} (9.20)

...
Proof of Theorem 9.3.3.

\[ L - H_D(X) = \sum_i p_i \ell_i - \sum_i p_i \log_D 1/p_i \]  \hspace{1cm} (9.15)

\[ = - \sum_i p_i \log_D D^{-\ell_i} + \sum_i p_i \log_D p_i \]  \hspace{1cm} (9.16)

\[ = - \sum_i p_i \log_D D^{-\ell_i} \]  \hspace{1cm} (9.17)

\[ + \sum_i p_i \log_D p_i \]  \hspace{1cm} (9.18)

\[ \geq 0 \quad \text{since} \quad c \leq 1 \]  \hspace{1cm} (9.20)

...
Optimal Code Lengths

Proof of Theorem 9.3.3.

\[ L - H_D(X) = \sum_i p_i \ell_i - \sum_i p_i \log_D 1/p_i \]  \hspace{1cm} (9.15)

\[ = - \sum_i p_i \log_D D^{-\ell_i} + \sum_i p_i \log_D p_i \]  \hspace{1cm} (9.16)

\[ = - \sum_i p_i \log_D D^{-\ell_i} + \log_D \left( \sum_i D^{-\ell_i} \right) - \log_D \left( \sum_i D^{-\ell_i} \right) \]  \hspace{1cm} (9.17)

\[ + \sum_i p_i \log_D p_i \]  \hspace{1cm} (9.18)

\[ (9.20) \]

...
Optimal Code Lengths

Proof of Theorem 9.3.3.

\[ L - H_D(X) = \sum_i p_i \ell_i - \sum_i p_i \log_D 1/p_i \]  \hspace{1cm} (9.15)

\[ = - \sum_i p_i \log_D D^{-\ell_i} + \sum_i p_i \log_D p_i \]  \hspace{1cm} (9.16)

\[ = - \sum_i p_i \log_D D^{-\ell_i} + \log_D\left(\sum_i D^{-\ell_i}\right) - \log_D\left(\sum_i D^{-\ell_i}\right) \]  \hspace{1cm} (9.17)

\[ + \sum_i p_i \log_D p_i \quad \text{(now define } r_i = \frac{D^{-\ell_i}}{\sum_i D^{-\ell_i}}) \]  \hspace{1cm} (9.18)

\[ \geq 0 \]  \hspace{1cm} (9.20)

...
Proof of Theorem 9.3.3.

\[ L - H_D(X) = \sum_i p_i \ell_i - \sum_i p_i \log_D 1/p_i \]  \hspace{1cm} (9.15)

\[ = - \sum_i p_i \log_D D^{-\ell_i} + \sum_i p_i \log_D p_i \]  \hspace{1cm} (9.16)

\[ = - \sum_i p_i \log_D D^{-\ell_i} + \log_D(\sum_i D^{-\ell_i}) - \log_D(\sum_i D^{-\ell_i}) \]  \hspace{1cm} (9.17)

\[ + \sum_i p_i \log_D p_i \]  \hspace{1cm} (now define \( r_i = \frac{D^{-\ell_i}}{\sum_i D^{-\ell_i}} \))  \hspace{1cm} (9.18)

\[ = \sum_i p_i \log \frac{p_i}{r_i} - \log_D(\sum_i D^{-\ell_i}) \]  \hspace{1cm} (9.20)
Proof of Theorem 9.3.3.

\[ L - H_D(X) = \sum_i p_i \ell_i - \sum_i p_i \log_D 1/p_i \quad (9.15) \]

\[ = - \sum_i p_i \log_D D^{-\ell_i} + \sum_i p_i \log_D p_i \quad (9.16) \]

\[ = - \sum_i p_i \log_D D^{-\ell_i} + \log_D \left( \sum_i D^{-\ell_i} \right) - \log_D \left( \sum_i D^{-\ell_i} \right) \quad (9.17) \]

\[ + \sum_i p_i \log_D p_i \quad \text{(now define } r_i = \frac{D^{-\ell_i}}{\sum_i D^{-\ell_i}}) \quad (9.18) \]

\[ = \sum_i p_i \log \frac{p_i}{r_i} - \log_D \left( \sum_i D^{-\ell_i} \right) = D(p||r) + \log_D(1/c) \quad (9.19) \]

\[ \geq 0 \quad \text{since } c \leq 1 \quad \text{by Kraft, where } c = \sum_i D^{-\ell_i} \quad (9.20) \]

\[ \ldots \]
Proof of Theorem 9.3.3.

\[
L - H_D(X) = \sum_i p_i \ell_i - \sum_i p_i \log_D 1/p_i
\]  
(9.15)

\[
= - \sum_i p_i \log_D D^{-\ell_i} + \sum_i p_i \log_D p_i
\]  
(9.16)

\[
= - \sum_i p_i \log_D D^{-\ell_i} + \log_D (\sum_i D^{-\ell_i}) - \log_D (\sum_i D^{-\ell_i})
\]  
(9.17)

\[
+ \sum_i p_i \log_D p_i \quad \text{(now define } r_i = \frac{D^{-\ell_i}}{\sum_i D^{-\ell_i}}) \]
(9.18)

\[
= \sum_i p_i \log \frac{p_i}{r_i} - \log_D (\sum_i D^{-\ell_i}) = D(p||r) + \log_D (1/c)
\]  
(9.19)

\[
\geq 0 \quad \text{since } c \leq 1 \text{ by Kraft, where } c = \sum_i D^{-\ell_i}
\]  
(9.20)

...
Proof of Theorem 9.3.3.

So we have that $L \geq H_D(X)$. 

Definition 9.3.4 (D-adic)

A probability distribution is called $D$-adic w.r.t. $D$ if each of the probabilities is $D^{-n}$ for some $n$.

Ex: when $D = 2$, the distribution $[\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}] = [2^{-1}, 2^{-2}, 2^{-3}, 2^{-3}]$ is 2-adic.

Thus, we have equality above iff the distribution is appropriately $D$-adic.
Optimal Code Lengths

...Proof of Theorem 9.3.3.

- So we have that $L \geq H_D(X)$.
- Equality, $L = H$ is achieved iff $p_i = D^{-\ell_i}$ for all $i \iff -\log_D p_i$ is an integer . . .
Optimal Code Lengths

... Proof of Theorem 9.3.3.

- So we have that \( L \geq H_D(X) \).
- Equality, \( L = H \) is achieved iff \( p_i = D^{-\ell_i} \) for all \( i \) \( \Leftrightarrow \) \( -\log_D p_i \) is an integer . . .
- . . . in which case \( c = \sum_i D^{-\ell_i} = 1 \)
Optimal Code Lengths

... Proof of Theorem 9.3.3.

- So we have that \( L \geq H_D(X) \).
- Equality, \( L = H \) is achieved iff \( p_i = D^{-\ell_i} \) for all \( i \) \( \Leftrightarrow \) \( -\log_D p_i \) is an integer ...
- ... in which case \( c = \sum_i D^{-\ell_i} = 1 \)
Proof of Theorem 9.3.3.

- So we have that $L \geq H_D(X)$.
- Equality, $L = H$ is achieved iff $p_i = D^{-\ell_i}$ for all $i \Leftrightarrow -\log_D p_i$ is an integer . . .
- . . . in which case $c = \sum_i D^{-\ell_i} = 1$

Definition 9.3.4 ($D$-adic)

A probability distribution is called $D$-adic w.r.t. $D$ if each of the probabilities is $= D^{-n}$ for some $n$. 
Optimal Code Lengths

...Proof of Theorem 9.3.3.

- So we have that $L \geq H_D(X)$.
- Equality, $L = H$ is achieved iff $p_i = D^{-\ell_i}$ for all $i \iff -\log_D p_i$ is an integer ...
- ...in which case $c = \sum_i D^{-\ell_i} = 1$

Definition 9.3.4 ($D$-adic)

A probability distribution is called $D$-adic w.r.t. $D$ if each of the probabilities is $= D^{-n}$ for some $n$.

- Ex: when $D = 2$, the distribution $[\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}] = [2^{-1}, 2^{-2}, 2^{-3}, 2^{-3}]$ is 2-adic.
Optimal Code Lengths

... Proof of Theorem 9.3.3.

- So we have that $L \geq H_D(X)$.
- Equality, $L = H$ is achieved iff $p_i = D^{-\ell_i}$ for all $i \Leftrightarrow -\log_D p_i$ is an integer ...
- ... in which case $c = \sum_i D^{-\ell_i} = 1$

Definition 9.3.4 ($D$-adic)

A probability distribution is called $D$-adic w.r.t. $D$ if each of the probabilities is $= D^{-n}$ for some $n$.

- Ex: when $D = 2$, the distribution $[\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}] = [2^{-1}, 2^{-2}, 2^{-3}, 2^{-3}]$ is 2-adic.
- Thus, we have equality above iff the distribution is appropriately $D$-adic.
To summarize the conditions and relations:

- In the relaxed optimization problem, we relaxed the lengths so that they need not be integers, and we get \( E\ell = H \).
Summary of recent results

To summarize the conditions and relations:

- In the relaxed optimization problem, we relaxed the lengths so that they need not be integers, and we get $E\ell = H$.
- Assuming Kraft is true (and thus a prefix code exists), we have (with integer lengths) that $E\ell \geq H$. 
Summary of recent results

To summarize the conditions and relations:

- In the relaxed optimization problem, we relaxed the lengths so that they need not be integers, and we get $E\ell = H$.
- Assuming Kraft is true (and thus a prefix code exists), we have (with integer lengths) that $E\ell \geq H$.
- I.e., if we assume Kraft, and $\ell_i = -\log_D p_i$ is an integer, then $E\ell = H$. 
Summary of recent results

To summarize the conditions and relations:

- In the relaxed optimization problem, we relaxed the lengths so that they need not be integers, and we get $E\ell = H$.
- Assuming Kraft is true (and thus a prefix code exists), we have (with integer lengths) that $E\ell \geq H$.
- I.e., if we assume Kraft, and $\ell_i = -\log D p_i$ is an integer, then $E\ell = H$.
- I.e., if we assume Kraft, and $\ell_i \neq -\log D p_i$, but the lengths $\ell_i$ are still integers, then we have $E\ell > H$ strictly.
Shannon Codes

- \( L - H = D(p||r) + \log_D 1/c \), with \( c = \sum_i D^{-\ell_i} \)
Shannon Codes

- \( L - H = D(p||r) + \log_D 1/c \), with \( c = \sum_i D^{-\ell_i} \)

- Thus, to produce a code, we find closest (in the KL sense) \( D \)-adic distribution w.r.t. \( D \) to \( p \) and then construct the code as in the proof of the Kraft inequality converse.
Shannon Codes

- $L - H = D(p||r) + \log_D 1/c$, with $c = \sum_i D^{-\ell_i}$

Thus, to produce a code, we find closest (in the KL sense) $D$-adic distribution w.r.t. $D$ to $p$ and then construct the code as in the proof of the Kraft inequality converse.

- In general, however, unless P=NP, it is hard to find the KL closest $D$-adic distribution (integer programming problem).
Shannon Codes

- \( L - H = D(p\|r) + \log_D 1/c \), with \( c = \sum_i D^{-\ell_i} \)
- Thus, to produce a code, we find closest (in the KL sense) \( D \)-adic distribution w.r.t. \( D \) to \( p \) and then construct the code as in the proof of the Kraft inequality converse.
- In general, however, unless \( P=NP \), it is hard to find the KL closest \( D \)-adic distribution (integer programming problem).
- **Shannon codes**: consider \( \ell_i = \lceil \log_D 1/p_i \rceil \) as the code lengths
Shannon Codes

$L - H = D(p||r) + \log_D 1/c$, with $c = \sum_i D^{-\ell_i}$

Thus, to produce a code, we find closest (in the KL sense) $D$-adic
distribution w.r.t. $D$ to $p$ and then construct the code as in the
proof of the Kraft inequality converse.

In general, however, unless $P=NP$, it is hard to find the KL closest
$D$-adic distribution (integer programming problem).

Shannon codes: consider $\ell_i = [\log_D 1/p_i]$ as the code lengths

$\sum_i D^{-\ell_i}$
Shannon Codes

- $L - H = D(p||r) + \log_D 1/c$, with $c = \sum_i D^{-\ell_i}$

- Thus, to produce a code, we find closest (in the KL sense) $D$-adic distribution w.r.t. $D$ to $p$ and then construct the code as in the proof of the Kraft inequality converse.

- In general, however, unless $P=NP$, it is hard to find the KL closest $D$-adic distribution (integer programming problem).

- **Shannon codes**: consider $\ell_i = \lceil \log_D 1/p_i \rceil$ as the code lengths

- $\sum_i D^{-\ell_i} = \sum_i D^{-\lceil \log 1/p_i \rceil}$
Shannon Codes

- $L - H = D(p||r) + \log_D 1/c$, with $c = \sum_i D^{-\ell_i}$

- Thus, to produce a code, we find closest (in the KL sense) $D$-adic distribution w.r.t. $D$ to $p$ and then construct the code as in the proof of the Kraft inequality converse.

- In general, however, unless $P=NP$, it is hard to find the KL closest $D$-adic distribution (integer programming problem).

- **Shannon codes**: consider $\ell_i = \lceil \log_D 1/p_i \rceil$ as the code lengths

- $\sum_i D^{-\ell_i} = \sum_i D^{-\lceil \log 1/p_i \rceil} \leq \sum_i D^{-\log 1/p_i}$
Shannon Codes

- \( L - H = D(p||r) + \log_D 1/c \), with \( c = \sum_i D^{-\ell_i} \)

- Thus, to produce a code, we find closest (in the KL sense) \( D \)-adic distribution w.r.t. \( D \) to \( p \) and then construct the code as in the proof of the Kraft inequality converse.

- In general, however, unless \( \text{P}=\text{NP} \), it is hard to find the KL closest \( D \)-adic distribution (integer programming problem).

- **Shannon codes**: consider \( \ell_i = \lceil \log_D 1/p_i \rceil \) as the code lengths

\[
\sum_i D^{-\ell_i} = \sum_i D^{- \lceil \log 1/p_i \rceil} \leq \sum_i D^{- \log 1/p_i} = \sum_i p_i
\]
Shannon Codes

- \( L - H = D(p||r) + \log_D 1/c \), with \( c = \sum_i D^{-\ell_i} \)
- Thus, to produce a code, we find closest (in the KL sense) \( D \)-adic distribution w.r.t. \( D \) to \( p \) and then construct the code as in the proof of the Kraft inequality converse.
- In general, however, unless \( P=NP \), it is hard to find the KL closest \( D \)-adic distribution (integer programming problem).
- **Shannon codes**: consider \( \ell_i = \lceil \log_D 1/p_i \rceil \) as the code lengths
- \( \sum_i D^{-\ell_i} = \sum_i D^{-\lceil \log 1/p_i \rceil} \leq \sum_i D^{-\log 1/p_i} = \sum_i p_i = 1 \)
Shannon Codes

- \( L - H = D(p||r) + \log_D 1/c \), with \( c = \sum_i D^{-\ell_i} \)
- Thus, to produce a code, we find closest (in the KL sense) \( D \)-adic distribution w.r.t. \( D \) to \( p \) and then construct the code as in the proof of the Kraft inequality converse.
- In general, however, unless P=NP, it is hard to find the KL closest \( D \)-adic distribution (integer programming problem).
- **Shannon codes**: consider \( \ell_i = \lceil \log_D 1/p_i \rceil \) as the code lengths
- \( \sum_i D^{-\ell_i} = \sum_i D^{-\lceil \log 1/p_i \rceil} \leq \sum_i D^{-\log 1/p_i} = \sum_i p_i = 1 \)
- This means Kraft inequality holds for these lengths, so there is a prefix code
Shannon Codes

- \( L - H = D(p||r) + \log_D 1/c \), with \( c = \sum_i D^{-\ell_i} \)
- Thus, to produce a code, we find closest (in the KL sense) \( D \)-adic distribution w.r.t. \( D \) to \( p \) and then construct the code as in the proof of the Kraft inequality converse.
- In general, however, unless \( P=NP \), it is hard to find the KL closest \( D \)-adic distribution (integer programming problem).
- Shannon codes: consider \( \ell_i = \lceil \log_D 1/p_i \rceil \) as the code lengths
- \( \sum_i D^{-\ell_i} = \sum_i D^{-\lceil \log 1/p_i \rceil} \leq \sum_i D^{-\log 1/p_i} = \sum_i p_i = 1 \)
- This means Kraft inequality holds for these lengths, so there is a prefix code (if the lengths were too short there might be a problem but we’re rounding up).
Shannon Codes

- \( L - H = D(p||r) + \log_D 1/c \), with \( c = \sum_i D^{-\ell_i} \)
- Thus, to produce a code, we find closest (in the KL sense) \( D \)-adic distribution w.r.t. \( D \) to \( p \) and then construct the code as in the proof of the Kraft inequality converse.
- In general, however, unless \( P=NP \), it is hard to find the KL closest \( D \)-adic distribution (integer programming problem).
- **Shannon codes**: consider \( \ell_i = \lceil \log_D 1/p_i \rceil \) as the code lengths
- \( \sum_i D^{-\ell_i} = \sum_i D^{-\lceil \log 1/p_i \rceil} \leq \sum_i D^{-\log 1/p_i} = \sum_i p_i = 1 \)
- This means Kraft inequality holds for these lengths, so there is a prefix code (if the lengths were too short there might be a problem but we’re rounding up).
- Also, we have a bound on lengths in terms of real numbers

\[
\log_D \frac{1}{p_i} \leq \ell_i < \log_D \frac{1}{p_i} + 1 \quad (9.21)
\]
Shannon Codes

- Taking expected values on both sides yields

\[ H_D(X) \leq L < H_D(X) + 1 \]  \hspace{1cm} (9.22)
Shannon Codes

- Taking expected values on both sides yields

\[ H_D(X) \leq L < H_D(X) + 1 \]  

(9.22)

- Close to the entropy, only one extra bit!
Taking expected values on both sides yields

\[
H_D(X) \leq L < H_D(X) + 1
\]  \hspace{1cm} (9.22)

Close to the entropy, only one extra bit! Is this good?
Shannon Codes

- Taking expected values on both sides yields

\[ H_D(X) \leq L < H_D(X) + 1 \]  \hspace{1cm} (9.22)

- Close to the entropy, only one extra bit! Is this good?
- Also, \( H \leq L^* \leq L \) where \( L^* \) is the optimal length for integer length codes (i.e., might not have optimal integer lengths satisfying Kraft).
Taking expected values on both sides yields

\[ H_D(X) \leq L < H_D(X) + 1 \]  

(9.22)

Close to the entropy, only one extra bit! Is this good?

Also, \( H \leq L^* \leq L \) where \( L^* \) is the optimal length for integer length codes (i.e., might not have optimal integer lengths satisfying Kraft).
Taking expected values on both sides yields

\[ H_D(X) \leq L < H_D(X) + 1 \] (9.22)

Close to the entropy, only one extra bit! Is this good?

Also, \( H \leq L^* \leq L \) where \( L^* \) is the optimal length for integer length codes (i.e., might not have optimal integer lengths satisfying Kraft).

**Theorem 9.4.1**

Let \( \ell_1^*, \ell_2^*, \ldots, \ell_m^* \) be the optimal integral codeword lengths for source \( p \) and \( D \)-ary alphabet. \( L^* \) is the expected length. Then

\[ H_D(X) \leq L^* < H_D(X) + 1 \] (9.23)
Shannon Codes

- Taking expected values on both sides yields

\[ H_D(X) \leq L < H_D(X) + 1 \]  \hspace{1cm} (9.22)

- Close to the entropy, only one extra bit! Is this good?
- Also, \( H \leq L^* \leq L \) where \( L^* \) is the optimal length for integer length codes (i.e., might not have optimal integer lengths satisfying Kraft).

**Theorem 9.4.1**

Let \( \ell_1^*, \ell_2^*, \ldots, \ell_m^* \) be the optimal integral codeword lengths for source \( p \) and \( D \)-ary alphabet. \( L^* \) is the expected length. Then

\[ H_D(X) \leq L^* < H_D(X) + 1 \]  \hspace{1cm} (9.23)

- So average overhead of using integers (rather than fractional) codeword lengths is no more than one bit per symbol.
How bad is one bit?

- How bad is this overhead?
How bad is one bit?

- How bad is this overhead?
- Depends on $H$. Efficiency of code

\[
0 \leq \text{Efficiency} \triangleq \frac{H_D(X)}{E\ell(X)} \leq 1 \quad (9.24)
\]
How bad is one bit?

- How bad is this overhead?
- Depends on $H$. Efficiency of code

$$0 \leq \text{Efficiency} \triangleq \frac{H_D(X)}{E\ell(X)} \leq 1$$  \hspace{1cm} (9.24)

- If $E\ell(X) = H_D(X) + 1$, then efficiency $\to 1$ as $H(X) \to \infty$. 

For small alphabets (or low-entropy distributions, such as close to deterministic distributions), impossible to have good efficiency.
How bad is one bit?

- How bad is this overhead?
- Depends on $H$. Efficiency of code

$$0 \leq \text{Efficiency} \triangleq \frac{H_D(X)}{E\ell(X)} \leq 1 \quad (9.24)$$

- If $E\ell(X) = H_D(X) + 1$, then efficiency $\rightarrow 1$ as $H(X) \rightarrow \infty$.
- Efficiency $\rightarrow 0$ as $H(X) \rightarrow 0$, so entropy would need to be very large for this to be good.
How bad is one bit?

- How bad is this overhead?
- Depends on $H$. Efficiency of code

$$0 \leq \text{Efficiency} \triangleq \frac{H_D(X)}{\mathbb{E}\ell(X)} \leq 1 \quad (9.24)$$

- If $\mathbb{E}\ell(X) = H_D(X) + 1$, then efficiency $\to 1$ as $H(X) \to \infty$.
- Efficiency $\to 0$ as $H(X) \to 0$, so entropy would need to be very large for this to be good.
- For small alphabets (or low-entropy distributions, such as close to deterministic distributions), impossible to have good efficiency.
How bad is one bit?

- How bad is this overhead?
- Depends on $H$. Efficiency of code

$$0 \leq \text{Efficiency} \triangleq \frac{H_D(X)}{E\ell(X)} \leq 1 \quad (9.24)$$

- If $E\ell(X) = H_D(X) + 1$, then efficiency $\to 1$ as $H(X) \to \infty$.
- Efficiency $\to 0$ as $H(X) \to 0$, so entropy would need to be very large for this to be good.
- For small alphabets (or low-entropy distributions, such as close to deterministic distributions), impossible to have good efficiency. E.g., $\mathcal{D} = \{0, 1\}$ then $\max H(X) = 1$, so best possible efficiency is 50% 😞.
Improving efficiency

- Such symbol codes are inherently disadvantaged, unless their distributions are $D$-adic.
Improving efficiency

- Such symbol codes are inherently disadvantaged, unless their distributions are $D$-adic.
- We can reduce overhead (improve efficiency) by coding $> 1$ symbol at a time (block code, or a vector code, the symbol is the vector).
Improving efficiency

- Such symbol codes are inherently disadvantaged, unless their distributions are $D$-adic.
- We can reduce overhead (improve efficiency) by coding $> 1$ symbol at a time (block code, or a vector code, the symbol is the vector).
- Let $L_n$ be the expected length of $n$ symbols $x_{1:n}$.
Improving efficiency

- Such symbol codes are inherently disadvantaged, unless their distributions are $D$-adic.
- We can reduce overhead (improve efficiency) by coding $>1$ symbol at a time (block code, or a vector code, the symbol is the vector).
- Let $L_n$ be the expected length of $n$ symbols $x_{1:n}$. $L_n$ is the expected per-symbol length, when encoding $n$ symbols.
Improving efficiency

- Such symbol codes are inherently disadvantaged, unless their distributions are $D$-adic.
- We can reduce overhead (improve efficiency) by coding $> 1$ symbol at a time (block code, or a vector code, the symbol is the vector).
- Let $L_n$ be the expected length of $n$ symbols $x_{1:n}$. $L_n$ is the expected per-symbol length, when encoding $n$ symbols.

$$L_n = \frac{1}{n} \sum_{x_{1:n}} p(x_{1:n}) \ell(x_{1:n}) = \frac{1}{n} E\ell(x_{1:n}) \quad (9.25)$$
Improving efficiency

- Such symbol codes are inherently disadvantaged, unless their distributions are $D$-adic.
- We can reduce overhead (improve efficiency) by coding $> 1$ symbol at a time (block code, or a vector code, the symbol is the vector).
- Let $L_n$ be the expected length of $n$ symbols $x_{1:n}$. $L_n$ is the expected per-symbol length, when encoding $n$ symbols.

$$ L_n = \frac{1}{n} \sum_{x_{1:n}} p(x_{1:n}) \ell(x_{1:n}) = \frac{1}{n} E\ell(x_{1:n}) $$

(9.25)

- Let's use Shannon coding lengths to get

$$ \log 1/p_i \leq \ell_i \leq \log 1/p_i + 1 $$

(9.26)
Improving efficiency

- Such symbol codes are inherently disadvantaged, unless their distributions are $D$-adic.
- We can reduce overhead (improve efficiency) by coding $>1$ symbol at a time (block code, or a vector code, the symbol is the vector).
- Let $L_n$ be the expected length of $n$ symbols $x_{1:n}$. $L_n$ is the expected per-symbol length, when encoding $n$ symbols.

$$L_n = \frac{1}{n} \sum_{x_{1:n}} p(x_{1:n}) \ell(x_{1:n}) = \frac{1}{n} E\ell(x_{1:n}) \quad (9.25)$$

- Let's use Shannon coding lengths to get

$$\sum_i p_i \left( \log \frac{1}{p_i} \leq \ell_i \leq \log \frac{1}{p_i} + 1 \right) \quad (9.26)$$

$$\Rightarrow H(X_1,\ldots,X_n) \leq E\ell(X_{1:n}) < H(X_1,\ldots,X_n) + 1 \quad (9.27)$$
Improving efficiency

- Such symbol codes are inherently disadvantaged, unless their distributions are $D$-adic.
- We can reduce overhead (improve efficiency) by coding $>1$ symbol at a time (block code, or a vector code, the symbol is the vector).
- Let $L_n$ be the expected length of $n$ symbols $x_{1:n}$. $L_n$ is the expected per-symbol length, when encoding $n$ symbols.

\[
L_n = \frac{1}{n} \sum_{x_{1:n}} p(x_{1:n}) \ell(x_{1:n}) = \frac{1}{n} E\ell(x_{1:n}) \tag{9.25}
\]

- Let's use Shannon coding lengths to get

\[
\sum_i p_i \left( \log \frac{1}{p_i} \leq \ell_i \leq \log \frac{1}{p_i} + 1 \right) \tag{9.26}
\]

\[
\Rightarrow H(X_1, \ldots, X_n) \leq E\ell(X_{1:n}) < H(X_1, \ldots, X_n) + 1 \tag{9.27}
\]
Improving efficiency

- If the $X_i$ are i.i.d. then $H(X_1, \ldots, X_n) = nH(X_i)$. 
Improving efficiency

- If the $X_i$ are i.i.d. then $H(X_1, \ldots, X_n) = nH(X_i)$.
- \( \Rightarrow \) we have that

\[
H(X) \leq L_n \leq H(X) + \frac{1}{n}
\]  

(9.28)
Improving efficiency

- If the $X_i$ are i.i.d. then $H(X_1, \ldots, X_n) = nH(X_i)$.
- $\Rightarrow$ we have that

\[ H(X) \leq L_n \leq H(X) + \frac{1}{n} \quad (9.28) \]

- As $n$ gets big, per symbol penalty of a Shannon code decreases,
Improving efficiency

- If the $X_i$ are i.i.d. then $H(X_1, \ldots, X_n) = n H(X_i)$.
- $\Rightarrow$ we have that

$$H(X) \leq L_n \leq H(X) + \frac{1}{n} \quad (9.28)$$

- As $n$ gets big, per symbol penalty of a Shannon code decreases, and we approach the Entropy limit (per symbol),
If the $X_i$ are i.i.d. then $H(X_1, \ldots, X_n) = nH(X_i)$.

⇒ we have that

$$H(X) \leq L_n \leq H(X) + \frac{1}{n} \quad (9.28)$$

As $n$ gets big, per symbol penalty of a Shannon code decreases, and we approach the Entropy limit (per symbol), although once again we have to code a block at a time.
Improving efficiency

- If the $X_i$ are i.i.d. then $H(X_1, \ldots, X_n) = nH(X_i)$.
- $\Rightarrow$ we have that

  $$H(X) \leq L_n \leq H(X) + \frac{1}{n} \quad (9.28)$$

- As $n$ gets big, per symbol penalty of a Shannon code decreases, and we approach the Entropy limit (per symbol), although once again we have to code a block at a time.
- Again, even if symbols are independent it is better to code jointly.
Stochastic processes

Consider any stationary (ergodic) stochastic process. Then

\[ H(X_1, \ldots, X_n) \leq E\ell(X_1, \ldots, X_n) < H(X_1, \ldots, X_n) + 1 \]  \hspace{1cm} (9.29)

\[ \Rightarrow H(X_1, \ldots, X_n) \leq \frac{1}{n} \sum_{n} \leq \ell(X_1, \ldots, X_n) < H(X_1, \ldots, X_n) + \frac{1}{n} \] \hspace{1cm} (9.30)
Consider any stationary (ergodic) stochastic process. Then

\[ H(X_1, \ldots, X_n) \leq E\ell(X_1, \ldots, X_n) < H(X_1, \ldots, X_n) + 1 \]  \hspace{1cm} (9.29)

\[ \Rightarrow \frac{H(X_1, \ldots, X_n)}{n} \leq L_n < \frac{H(X_1, \ldots, X_n)}{n} + \frac{1}{n} \]  \hspace{1cm} (9.30)
Stochastic processes

Consider any stationary (ergodic) stochastic process. Then

\[ H(X_1, \ldots, X_n) \leq E\ell(X_1, \ldots, X_n) < H(X_1, \ldots, X_n) + 1 \]  \hspace{1cm} (9.29)

\[ \Rightarrow \frac{H(X_1, \ldots, X_n)}{n} \leq L_n < \frac{H(X_1, \ldots, X_n)}{n} + \frac{1}{n} \]  \hspace{1cm} (9.30)

If stationary, then l.h.s. \( \to H(\mathcal{X}) \) as \( n \to \infty \).
Consider any stationary (ergodic) stochastic process. Then
\[ H(X_1, \ldots, X_n) \leq E\ell(X_1, \ldots, X_n) < H(X_1, \ldots, X_n) + 1 \] (9.29)

\[ \Rightarrow \frac{H(X_1, \ldots, X_n)}{n} \leq L_n < \frac{H(X_1, \ldots, X_n)}{n} + \frac{1}{n} \] (9.30)

If stationary, then l.h.s. \( \rightarrow H(X) \) as \( n \rightarrow \infty \).

Thus, as \( n \) gets large, expected length of code goes to the entropy rate of the stochastic process.
Stochastic processes

Consider any stationary (ergodic) stochastic process. Then

\[ H(X_1, \ldots, X_n) \leq E\ell(X_1, \ldots, X_n) < H(X_1, \ldots, X_n) + 1 \]  \hspace{1cm} (9.29)

\[ \Rightarrow \frac{H(X_1, \ldots, X_n)}{n} \leq L_n < \frac{H(X_1, \ldots, X_n)}{n} + \frac{1}{n} \]  \hspace{1cm} (9.30)

- If stationary, then l.h.s. \( \rightarrow H(\mathcal{X}) \) as \( n \rightarrow \infty \).
- Thus, as \( n \) gets large, expected length of code goes to the entropy rate of the stochastic process.
- We can make penalty per source symbol as small as we want if we don’t mind long block lengths. This can be stated as a theorem.
Stochastic processes

- Consider any stationary (ergodic) stochastic process. Then
  \[ H(X_1, \ldots, X_n) \leq E\ell(X_1, \ldots, X_n) < H(X_1, \ldots, X_n) + 1 \]  
  \[ \Rightarrow \frac{H(X_1, \ldots, X_n)}{n} \leq L_n < \frac{H(X_1, \ldots, X_n)}{n} + \frac{1}{n} \]

- If stationary, then l.h.s. \( \to H(X) \) as \( n \to \infty \).
- Thus, as \( n \) gets large, expected length of code goes to the entropy rate of the stochastic process.
- We can make penalty per source symbol as small as we want if we don’t mind long block lengths. This can be stated as a theorem

Theorem 9.4.2

Minimum expected codeword lengths per symbol satisfy

\[ \frac{H(X_1, \ldots, X_n)}{n} \leq L^*_n < \frac{H(X_1, \ldots, X_n)}{n} + \frac{1}{n} \]

if \( X_i \) is stationary. I.e., \( L^* \to H(X) \)
Coding with the wrong distribution

- In general, we don't have the “true” distribution (if there is one).
Coding with the wrong distribution

- In general, we don’t have the “true” distribution (if there is one).
- With the wrong distribution, we’ll make mistakes.
Coding with the wrong distribution

- In general, we don’t have the “true” distribution (if there is one).
- With the wrong distribution, we’ll make mistakes. I.e., Shannon code would use lengths \( \ell(x) = \lceil \log \frac{1}{q(x)} \rceil \) but the true probability is \( p(x) \neq q(x) \).

\[
\text{(9.35)}
\]
Coding with the wrong distribution

- In general, we don’t have the “true” distribution (if there is one).
- With the wrong distribution, we’ll make mistakes. I.e., Shannon code would use lengths $\ell(x) = \lceil \log \frac{1}{q(x)} \rceil$ but the true probability is $p(x) \neq q(x)$. How does this hurt us?

\[
\mathbb{E}[\ell(X)] = \sum_x p(x) \cdot \lceil \log \frac{1}{q(x)} \rceil \leq \sum_x p(x) \cdot (\log \frac{1}{q(x)} + 1) = \sum_x p(x) \cdot \log \frac{p(x)}{q(x)} + \sum_x p(x) \cdot \log \frac{1}{p(x)} + 1 \tag{9.32}
\]

\[
= D(p||q) + H(p) + 1 \tag{9.33}
\]

Thus, $D(p||q)$ is per symbol bit penalty for using wrong distribution.
Coding with the wrong distribution

- In general, we don’t have the “true” distribution (if there is one).
- With the wrong distribution, we’ll make mistakes. I.e., Shannon code would use lengths $\ell(x) = \lceil \log \frac{1}{q(x)} \rceil$ but the true probability is $p(x) \neq q(x)$. How does this hurt us?

$$E\ell(X)$$

(9.35)
Coding with the wrong distribution

- In general, we don’t have the “true” distribution (if there is one).
- With the wrong distribution, we’ll make mistakes. I.e., Shannon code would use lengths \( \ell(x) = \lceil \log 1/q(x) \rceil \) but the true probability is \( p(x) \neq q(x) \). How does this hurt us?

\[
E\ell(X) = \sum_x p(x) \log 1/q(x)
\]  

(9.35)
Coding with the wrong distribution

- In general, we don’t have the “true” distribution (if there is one).
- With the wrong distribution, we’ll make mistakes. I.e., Shannon code would use lengths \( \ell(x) = \lceil \log 1/q(x) \rceil \) but the true probability is \( p(x) \neq q(x) \). How does this hurt us?

\[
E\ell(X) = \sum_x p(x) \lceil \log 1/q(x) \rceil \leq \sum_x p(x) (\log \frac{1}{q(x)} + 1) \tag{9.32}
\]

\[
D(p||q) + H(p) + 1 \tag{9.35}
\]
Coding with the wrong distribution

- In general, we don’t have the “true” distribution (if there is one).
- With the wrong distribution, we’ll make mistakes. I.e., Shannon code would use lengths \( \ell(x) = \lceil \log \frac{1}{q(x)} \rceil \) but the true probability is \( p(x) \neq q(x) \). How does this hurt us?

\[
E\ell(X) = \sum_x p(x) \lceil \log \frac{1}{q(x)} \rceil \leq \sum_x p(x) \left( \log \frac{1}{q(x)} + 1 \right) \quad (9.32)
\]

\[
= \sum_x p(x) \left( \log \frac{p(x)}{q(x)} \frac{1}{p(x)} + 1 \right) \quad (9.33)
\]

\[
= D(p || q) + H(p) + 1 \quad (9.35)
\]
Coding with the wrong distribution

- In general, we don’t have the “true” distribution (if there is one).
- With the wrong distribution, we’ll make mistakes. I.e., Shannon code would use lengths \( \ell(x) = \lceil \log \frac{1}{q(x)} \rceil \) but the true probability is \( p(x) \neq q(x) \). How does this hurt us?

\[
E\ell(X) = \sum_x p(x) \lceil \log \frac{1}{q(x)} \rceil \leq \sum_x p(x) (\log \frac{1}{q(x)} + 1) \quad (9.32)
\]
\[
= \sum_x p(x) (\log \frac{p(x)}{q(x)} \frac{1}{p(x)} + 1) \quad (9.33)
\]
\[
= \sum_x p(x) \log \frac{p(x)}{q(x)} + \sum_x p(x) \log \frac{1}{p(x)} + 1 \quad (9.34)
\]
\[
= D(p||q) + H(p) + 1 \quad (9.35)
\]
Coding with the wrong distribution

- In general, we don’t have the “true” distribution (if there is one).
- With the wrong distribution, we’ll make mistakes. I.e., Shannon code would use lengths $\ell(x) = \lceil \log \frac{1}{q(x)} \rceil$ but the true probability is $p(x) \neq q(x)$. How does this hurt us?

\[
E\ell(X) = \sum_{x} p(x) \lceil \log \frac{1}{q(x)} \rceil \leq \sum_{x} p(x) \left( \log \frac{1}{q(x)} + 1 \right) \quad (9.32)
\]

\[
= \sum_{x} p(x) \left( \log \frac{p(x)}{q(x)} \frac{1}{p(x)} + 1 \right) \quad (9.33)
\]

\[
= \sum_{x} p(x) \log \frac{p(x)}{q(x)} + \sum_{x} p(x) \log \frac{1}{p(x)} + 1 \quad (9.34)
\]

\[
= D(p||q) + H(p) + 1 \quad (9.35)
\]
Coding with the wrong distribution

- In general, we don’t have the “true” distribution (if there is one).
- With the wrong distribution, we’ll make mistakes. I.e., Shannon code would use lengths $\ell(x) = \lceil \log \frac{1}{q(x)} \rceil$ but the true probability is $p(x) \neq q(x)$. How does this hurt us?

\[
E\ell(X) = \sum_x p(x) \lceil \log \frac{1}{q(x)} \rceil \leq \sum_x p(x) (\log \frac{1}{q(x)} + 1) \quad (9.32)
\]

\[
= \sum_x p(x) (\log \frac{p(x)}{q(x)} \frac{1}{p(x)} + 1) \quad (9.33)
\]

\[
= \sum_x p(x) \log \frac{p(x)}{q(x)} + \sum_x p(x) \log \frac{1}{p(x)} + 1 \quad (9.34)
\]

\[
= D(p\|q) + H(p) + 1 \quad (9.35)
\]

- Thus, $D(p\|q)$ is per symbol bit penalty for using wrong distribution.
Coding with the wrong distribution

**Theorem 9.4.3**

*Expected length under* $p(x)$ *of code with* $\ell(x) = \lceil \log \frac{1}{q(x)} \rceil$ *satisfies*

\[
H(p) + D(p||q) \leq E_p \ell(X) \leq H(p) + D(p||q) + 1 \tag{9.36}
\]

- l.h.s. is the best we can do with the wrong distribution $q$ when the true distribution is $p$. 
Goal is to find a code with the shortest possible expected length.

From the above code class, we might think that we want to use codes from the largest class possible (since we might think we’re more likely to get shorter codes).
Kraft revisited

- We proved Kraft inequality is true for instantaneous codes (and vice versa).
Kraft revisited

- We proved Kraft inequality is true for instantaneous codes (and vice versa).
- Could it be true for all uniquely decodable codes?
Kraft revisited

- We proved Kraft inequality is true for instantaneous codes (and vice versa).
- Could it be true for all uniquely decodable codes?
- Could larger class of codes have shorter expected codeword lengths?
Kraft revisited

- We proved Kraft inequality is true for instantaneous codes (and vice versa).
- Could it be true for all uniquely decodable codes?
- Could larger class of codes have shorter expected codeword lengths?
- Since larger, we might (naïvely) expect that we could do better.
Kraft revisited

- We proved Kraft inequality is true for instantaneous codes (and vice versa).
- Could it be true for all uniquely decodable codes?
- Could larger class of codes have shorter expected codeword lengths?
- Since larger, we might (naïvely) expect that we could do better.

**Theorem 9.5.1**

Codeword lengths of any uniquely decodable code (not nec. instantaneous) must satisfy Kraft inequality \( \sum_i D^{-\ell_i} \leq 1. \)
Kraft revisited

- We proved Kraft inequality is true for instantaneous codes (and vice versa).
- Could it be true for all uniquely decodable codes?
- Could larger class of codes have shorter expected codeword lengths?
- Since larger, we might (naïvely) expect that we could do better.

**Theorem 9.5.1**

Codeword lengths of any uniquely decodable code (not nec. instantaneous) must satisfy Kraft inequality $\sum_i D^{-\ell_i} \leq 1$. Conversely, given a set of codeword lengths that satisfy Kraft, it is possible to construct a uniquely decodable code.
Kraft revisited

- We proved Kraft inequality is true for instantaneous codes (and vice versa).
- Could it be true for all uniquely decodable codes?
- Could larger class of codes have shorter expected codeword lengths?
- Since larger, we might (naïvely) expect that we could do better.

**Theorem 9.5.1**

Codeword lengths of any uniquely decodable code (not. nec. instantaneous) must satisfy Kraft inequality \( \sum_i D^{-\ell_i} \leq 1 \). Conversely, given a set of codeword lengths that satisfy Kraft, it is possible to construct a uniquely decodable code.

**Proof.**

Proof converse we already saw before (given lengths, we can construct a prefix code which is thus uniquely decodable). Thus we only need prove the first part.

...
Proof of Theorem 9.5.1.

- Given: uniquely decodable (not necessarily instantaneous) code with lengths \( \ell(x) \),
Proof of Theorem 9.5.1.

Given: uniquely decodable (not necessarily instantaneous) code with lengths $\ell(x)$, and length of $k$-extension $\ell(x, \ldots, x_k) = \sum_{i=1}^{k} \ell(x_i)$
Proof of Theorem 9.5.1.

Given: uniquely decodable (not necessarily instantaneous) code with lengths $\ell(x)$, and length of $k$-extension $\ell(x, \ldots, x_k) = \sum_{i=1}^{k} \ell(x_i)$ we wish to prove that $\sum_x D^{-\ell(x)} \leq 1$. 
Proof of Theorem 9.5.1.

- Given: uniquely decodable (not necessarily instantaneous) code with lengths $\ell(x)$, and length of $k$-extension $\ell(x, \ldots, x_k) = \sum_{i=1}^{k} \ell(x_i)$

  we wish to prove that $\sum_x D^{-\ell(x)} \leq 1$.

- Define $S = \sum_{x \in \mathcal{X}} D^{-\ell(x)}$,
Kraft and uniquely decodable

Proof of Theorem 9.5.1.

- Given: uniquely decodable (not necessarily instantaneous) code with lengths \( \ell(x) \), and length of \( k \)-extension \( \ell(x, \ldots, x_k) = \sum_{i=1}^{k} \ell(x_i) \)
  we wish to prove that \( \sum_x D^{-\ell(x)} \leq 1 \).

- Define \( S = \sum_{x \in \mathcal{X}} D^{-\ell(x)} \), then

\[
S_k = \sum_{x \in \mathcal{X}} D^{-\ell(x)} = \sum_{x_1: x_k \in \mathcal{X}} D^{-\ell(x_1: x_k)} \quad \text{(9.38)}
\]

\[
S_k = \sum_{x_1: x_k \in \mathcal{X}} D^{-\ell(x_1: x_k)} = \sum_{x_1: x_k \in \mathcal{X}} D^{-\ell(x_1)} - \ell(x_2) - \ldots - \ell(x_k) \quad \text{(9.39)}
\]
Kraft and uniquely decodable

Proof of Theorem 9.5.1.

- Given: uniquely decodable (not necessarily instantaneous) code with lengths \( \ell(x) \), and length of \( k \)-extension \( \ell(x, \ldots, x_k) = \sum_{i=1}^{k} \ell(x_i) \)
  we wish to prove that \( \sum_x D^{-\ell(x)} \leq 1 \).

- Define \( S = \sum_{x \in X} D^{-\ell(x)} \), then

\[
S^k
\]

(9.39)
Kraft and uniquely decodable

Proof of Theorem 9.5.1.

- Given: uniquely decodable (not necessarily instantaneous) code with lengths $\ell(x)$, and length of $k$-extension $\ell(x, \ldots, x_k) = \sum_{i=1}^{k} \ell(x_i)$ we wish to prove that $\sum_x D^{-\ell(x)} \leq 1$.
- Define $S = \sum_{x \in X} D^{-\ell(x)}$, then

$$S^k = \left[ \sum_x D^{-\ell(x)} \right]^k \quad (9.39)$$
Kraft and uniquely decodable

Proof of Theorem 9.5.1.

- Given: uniquely decodable (not necessarily instantaneous) code with lengths $\ell(x)$, and length of $k$-extension $\ell(x, \ldots, x_k) = \sum_{i=1}^{k} \ell(x_i)$ we wish to prove that $\sum_x D^{-\ell(x)} \leq 1$.
- Define $S = \sum_{x \in \mathcal{X}} D^{-\ell(x)}$, then

\[
S^k = \left[ \sum_x D^{-\ell(x)} \right]^k = \sum_{x_1:k \in \mathcal{X}^k} D^{-\ell(x_1)} D^{-\ell(x_2)} \ldots D^{-\ell(x_k)} \tag{9.37}
\]

\[
(9.39)
\]
Proof of Theorem 9.5.1.

- Given: uniquely decodable (not necessarily instantaneous) code with lengths \( \ell(x) \), and length of \( k \)-extension \( \ell(x, \ldots, x_k) = \sum_{i=1}^{k} \ell(x_i) \) we wish to prove that \( \sum x D^{-\ell(x)} \leq 1 \).

- Define \( S = \sum_{x \in \mathcal{X}} D^{-\ell(x)} \), then

\[
S^k = \left[ \sum_{x} D^{-\ell(x)} \right]^k = \sum_{x_1:k \in \mathcal{X}^k} D^{-\ell(x_1)} D^{-\ell(x_2)} \ldots D^{-\ell(x_k)} \quad (9.37)
\]

\[
= \sum_{x_1:k \in \mathcal{X}^k} D^{-[\sum_{i=1}^{k} \ell(x_i)]} \quad (9.39)
\]
Proof of Theorem 9.5.1.

- Given: uniquely decodable (not necessarily instantaneous) code with lengths $\ell(x)$, and length of $k$-extension $\ell(x, \ldots, x_k) = \sum_{i=1}^{k} \ell(x_i)$, we wish to prove that $\sum_x D^{-\ell(x)} \leq 1$.

- Define $S = \sum_{x \in X} D^{-\ell(x)}$, then

$$S^k = \left[ \sum_x D^{-\ell(x)} \right]^k = \sum_{x_{1:k} \in X^k} D^{-\ell(x_1)} D^{-\ell(x_2)} \ldots D^{-\ell(x_k)} \quad (9.37)$$

$$= \sum_{x_{1:k} \in X^k} D^{-[\sum_{i=1}^{k} \ell(x_i)]} = \sum_{x_{1:k} \in X^k} D^{-\ell(x_{1:k})} \quad (9.38)$$

$$\vdots$$
Kraft and uniquely decodable

Proof of Theorem 9.5.1.

- Given: uniquely decodable (not necessarily instantaneous) code with lengths $\ell(x)$, and length of $k$-extension $\ell(x, \ldots, x_k) = \sum_{i=1}^{k} \ell(x_i)$
- we wish to prove that $\sum_{x} D^{-\ell(x)} \leq 1$.
- Define $S = \sum_{x \in \mathcal{X}} D^{-\ell(x)}$, then

$$S^k = \left[ \sum_{x} D^{-\ell(x)} \right]^k = \sum_{x_1:k \in \mathcal{X}^k} D^{-\ell(x_1)} D^{-\ell(x_2)} \ldots D^{-\ell(x_k)} \quad (9.37)$$

$$= \sum_{x_1:k \in \mathcal{X}^k} D^{-[\sum_{i=1}^{k} \ell(x_i)]} = \sum_{x_1:k \in \mathcal{X}^k} D^{-\ell(x_1:k)} \quad (9.38)$$

$$= \sum_{m=1}^{k\ell\max} a(m) D^{-m} \quad (9.39)$$
Proof of Theorem 9.5.1.

\[ k\ell_{\text{max}} \sum_{m=1}^{\ell_{\text{max}}} a(m) D^{-m} \quad (9.39) \]

- where \( \ell_{\text{max}} = \max_x \ell(x) \) is the maximum codeword length.
Kraft and uniquely decodable

Proof of Theorem 9.5.1.

\[
\sum_{m=1}^{k \ell_{\text{max}}} a(m) D^{-m}
\]

where \( \ell_{\text{max}} = \max_x \ell(x) \) is the maximum codeword length.

- \( a(m) \) = number of source sequences \( x_{1:k} \) mapped into code words of length \( m \), i.e.,

\[
a(m) = \left| \left\{ x_{1:k} \in \mathcal{X}^k : \ell(x_{1:k}) = m \right\} \right|
\]
Kraft and uniquely decodable

Proof of Theorem 9.5.1.

\[
\sum_{m=1}^{k\ell_{\text{max}}} a(m) D^{-m}
\]  

(9.39)

- where \( \ell_{\text{max}} = \max_x \ell(x) \) is the maximum codeword length.
- \( a(m) = \) number of source sequences \( x_{1:k} \) mapped into code words of length \( m \), i.e.,

\[
a(m) = \left| \left\{ x_{1:k} \in \mathcal{X}^k : \ell(x_{1:k}) = m \right\} \right|
\]

(9.40)

- There are \( D^m \) codewords of length \( m \), and each of them can have (at most) one associated source sequence (since code is uniquely decodable).  

...
Proof of Theorem 9.5.1.

\[ \sum_{m=1}^{k \ell_{\text{max}}} a(m) D^{-m} \]  \hspace{1cm} (9.39)

- where \( \ell_{\text{max}} = \max_x \ell(x) \) is the maximum codeword length.
- \( a(m) = \) number of source sequences \( x_{1:k} \) mapped into code words of length \( m \), i.e.,

\[ a(m) = \left| \left\{ x_{1:k} \in X^k : \ell(x_{1:k}) = m \right\} \right| \]  \hspace{1cm} (9.40)

- There are \( D^m \) codewords of length \( m \), and each of them can have (at most) one associated source sequence (since code is uniquely decodable). Hence, \( a(m) \leq D^m \). 

...
Kraft and uniquely decodable

... proof of Theorem 9.5.1.

So continuing,

\[ S_k \leq k \ell \max \sum_{m=1}^{\infty} a_m D_m \leq k \ell \max \sum_{m=1}^{\infty} D_m D_m = k \ell \max \forall k \] (9.41)

So, \( S_k \) (exponential in \( k \)) never greater than \( k \ell \max \) (polynomial in \( k \)) \( \Rightarrow S_k \leq 1 \).

Giving \( S_k = \sum_{x \in X} D(x) - \ell(x) \leq 1 \).
proof of Theorem 9.5.1.

So continuing,

\[ S^k = \sum_{m=1}^{k \ell_{\text{max}}} a(m) D^{-m} \]  

(9.41)
Kraft and uniquely decodable

Proof of Theorem 9.5.1.

So continuing,

\[ S^k = \sum_{m=1}^{k\ell_{\text{max}}} a(m) D^{-m} \leq \sum_{m=1}^{k\ell_{\text{max}}} D^m D^{-m} \]

(9.41)
Kraft and uniquely decodable

... proof of Theorem 9.5.1.

So continuing,

\[ S^k = \sum_{m=1}^{k\ell_{\text{max}}} a(m) D^{-m} \leq \sum_{m=1}^{k\ell_{\text{max}}} D^m D^{-m} = k\ell_{\text{max}} \]  

(9.41)
Kraft and uniquely decodable

... proof of Theorem 9.5.1.

So continuing,

\[ S^k = \sum_{m=1}^{k\ell_{\text{max}}} a(m) D^{-m} \leq \sum_{m=1}^{k\ell_{\text{max}}} D^m D^{-m} = k\ell_{\text{max}} \quad \forall k \] (9.41)
Kraft and uniquely decodable

... proof of Theorem 9.5.1.

- So continuing,

\[ S^k = \sum_{m=1}^{k\ell_{\text{max}}} a(m)D^{-m} \leq \sum_{m=1}^{k\ell_{\text{max}}} D^m D^{-m} = k\ell_{\text{max}} \quad \forall k \quad (9.41) \]

- So, \( S^k \) (exponential in \( k \)) never greater than \( k\ell_{\text{max}} \) (polynomial in \( k \)) \( \Rightarrow S \leq 1. \)
Kraft and uniquely decodable

...proof of Theorem 9.5.1.

- So continuing,

\[ S^k = \sum_{m=1}^{k \ell_{\text{max}}} a(m) D^{-m} \leq \sum_{m=1}^{k \ell_{\text{max}}} D^m D^{-m} = k \ell_{\text{max}} \quad \forall k \quad (9.41) \]

- So, \( S^k \) (exponential in \( k \)) never greater than \( k \ell_{\text{max}} \) (polynomial in \( k \)) \( \Rightarrow S \leq 1 \).
- Giving \( S = \sum_{x \in \mathcal{X}} D^{-\ell(x)} \leq 1 \).
Summary: uniquely decodable vs. instantaneous codes

- Set of achievable codeword lengths the same for uniquely decodable codes and for instantaneous codes.
Summary: uniquely decodable vs. instantaneous codes

- Set of achievable codeword lengths the same for uniquely decodable codes and for instantaneous codes.
- $\Rightarrow$ optimal codeword length bound still holds.
Summary: uniquely decodable vs. instantaneous codes

- Set of achievable codeword lengths the same for uniquely decodable codes and for instantaneous codes.
- ⇒ optimal codeword length bound still holds.
- In fact, this is not surprising since we can get arbitrarily close to entropy rate already using instantaneous code (e.g., Shannon code) with long block words.
Summary: uniquely decodable vs. instantaneous codes

- Set of achievable codeword lengths the same for uniquely decodable codes and for instantaneous codes.
- ⇒ optimal codeword length bound still holds.
- In fact, this is not surprising since we can get arbitrarily close to entropy rate already using instantaneous code (e.g., Shannon code) with long block words.
- So, for distortionless symbol codes, we can then just consider instantaneous codes with impunity.
Summary: uniquely decodable vs. instantaneous codes

- Set of achievable codeword lengths the same for uniquely decodable codes and for instantaneous codes.
- \( \Rightarrow \) optimal codeword length bound still holds.
- In fact, this is not surprising since we can get arbitrarily close to entropy rate already using instantaneous code (e.g., Shannon code) with long block words.
- So, for distortionless symbol codes, we can then just consider instantaneous codes with impunity.
- Soon, we’ll talk about stream codes where we can get the benefit of long block lengths but we don’t have to wait for the end of a block before we start decoding, which is very useful for “streaming” applications like streaming audio/video.