Lecture 2:

Uniquely Decodable and Instantaneous Codes

Sam Roweis

September 15, 2005

Lossless Data Compression

- Let's focus on the lossless data compression problem for now, and not worry about noisy channel coding for now. In practice these two problems are handeled separately, i.e. we first design an efficient code for the source (removing source redundancy) and then (if necessary) we design a channel code to help us transmit the source code over the channel (adding redundancy).
- Assumptions (for now):
  1. the channel is perfectly noiseless
     i.e. the receiver sees exactly the encoder's output
  2. we always require the output of the decoder to exactly match the original sequence X.
  3. X is generated according to a fixed probabilistic model, $p(X)$
- We will measure the quality of our compression scheme (called a *code*) by examining the *average length* of the encoded string Z, averaged over $p(X)$.

Recall: Mathematical Setup

- Start with a sequence of symbols $X = X_1, X_2, \ldots, X_N$ from a finite source alphabet $\mathcal{A}_X = \{a_1, a_2, \ldots\}$.
- Examples: $\mathcal{A}_X = \{A,\ B,\ \ldots,\ Z,\ \_\}$; $\mathcal{A}_X = \{0,\ 1,\ 2,\ \ldots,\ 255\}$; $\mathcal{A}_X = \{C,\ G,\ T,\ A\}$; $\mathcal{A}_X = \{0,\ 1\}$.
- *Encoder*: outputs a new sequence $Z = Z_1, Z_2, \ldots, Z_M$ (using a possibly different code alphabet $\mathcal{A}_Z$).
- *Decoder* tries to convert Z back into X.
- In *compression*, the encoder tries to remove source redundancy.
- In *noisy channel coding*, the encoder tries to protect the message against transmission errors.
- We almost always use $\mathcal{A}_Z = 0, 1$ (e.g. computer files, digital communication) but the theory can be generalized to any finite set.

Encoding One Symbol at a Time

- To begin with, let's think about encoding one symbol X_i at a time, using a fixed code that defines a mapping of each source symbol into a finite sequence of code symbols called a *codeword*.
  (Later on we will consider encoding blocks of symbols together.)
- We will encode a sequence of source symbols X by concatenating the codewords of each.
- This is called a *symbol code*.
- E.g. source alphabet is $\mathcal{A}_X = \{C,\ G,\ T,\ A\}$. One possible code:
  $C \to 0; \quad G \to 10; \quad T \to 110; \quad A \to 1110$
  So we would have $CCAT \to 001110110$.
- We require that the mapping be such that we can *decode* this sequence, no matter what the original symbols were.

Notation for Sequences & Codes

- $\mathcal{A}_X$ and $\mathcal{A}_Z$ are the source and code alphabets.
- $\mathcal{A}_X^+$ and $\mathcal{A}_Z^+$ denote sequences of one or more symbols from the source or code alphabets.
- A symbol code, C, is a mapping $\mathcal{A}_X \to \mathcal{A}_Z^+$.
  We use $c(x)$ to denote the codeword to which C maps x.
- We use concatenation to extend this to a mapping for the *extended code*, $C^+ : \mathcal{A}_X^+ \to \mathcal{A}_Y^+$:
  $$c^+(x_1 x_2 \cdots x_N) = c(x_1)c(x_2)\cdots c(x_N)$$
  i.e., we code a string of symbols by just stringing together the codes for each symbol.
- I'll sometimes also use C to denote the set of all legal codewords: $\{w \mid w = C(a) \text{ for some } a \in \mathcal{A}_X\}$.

Uniquely Decodable & Instantaneous Codes

- A code is *uniquely decodable* if the mapping $C^+ : \mathcal{A}_X^+ \to \mathcal{A}_Z^+$ is one-to-one, i.e. $\forall\ \mathbf{x}$ and $\mathbf{x}'$ in $\mathcal{A}_X^+$, $\mathbf{x} \neq \mathbf{x}' \Rightarrow c^+(\mathbf{x}) \neq c^+(\mathbf{x}')$
- A code is obviously not uniquely decodable if two symbols have the same codeword — ie, if $c(a_i) = c(a_j)$ for some $i \neq j$ — so we'll usually assume that this isn't the case.
- A code is *instantaneously decodable* if any source sequences $\mathbf{x}$ and $\mathbf{x}'$ in $\mathcal{A}^+$ for which $\mathbf{x}$ is not a prefix of $\mathbf{x}'$ have encodings $\mathbf{z} = C(\mathbf{x})$ and $\mathbf{z}' = C(\mathbf{x}')$ for which $\mathbf{z}$ is not a prefix of $\mathbf{z}'$.
  Otherwise, after receiving $\mathbf{z}$, we wouldn't yet know whether the message starts with $\mathbf{z}$ or with $\mathbf{z}'$.
- Instantaneous codes are also called *prefix-free codes* or just *prefix codes*.

What Codes are Decodable?

- We only want to consider codes that can be successfully decoded.
- To define what that means, we need to set some rules of the game:
  1. How does the channel terminate the transmission?
     (e.g. it could explicitly mark the end, it could send only 0s after the end, it could send random garbage after the end,...)
  2. How soon do we require a decoded symbol to be known?
     (e.g. "instantaneously" – as soon as the codeword for the symbol is received, within a fixed delay of when its codeword is received, not until the entire message has been received,...)
- Easiest case: assume the end of the transmission is explicitly marked, and don't require any symbols to be decoded until the entire transmission has been received.
- Hardest case: require instantaneous decoding, and thus it doesn't matter what happens at the end of the transmission.

Examples

	Code A	Code B	Code C	Code D
a	10	0	0	0
b	11	10	01	01
c	111	110	011	11

Code A: Not uniquely decodable
Both bbb and cc encode as 111111

Code B: Instantaneously decodable
End of each codeword marked by 0

Code C: Decodable with one-symbol delay
End of codeword marked by *following* 0

Code D: Uniquely decodable, but with unbounded delay:
011111111111111 decodes as $accccccc$
01111111111111 decodes as $bcccccc$

More Examples

	Code E	Code F	Code G
a	100	0	0
b	101	001	01
c	010	010	011
d	011	100	1110

Code E: Instantaneously decodable
All codewords same length

Code F: Not uniquely decodable
e.g. baa,aca,aad all encode as 00100

Code G: Decodable with six-symbol delay.
(Try to work out why.)

A Check for Unique Decodability

- The *Sardinas-Patterson Theorem* tells us how to check whether a code is uniquely decodable.
- Let C be the set of codewords. Define $C_0 = C$.
  For $n > 0$, define
  $$C_n = \{w \in \mathcal{A}_X^+ \mid uw = v \text{ where } u \in C,\ v \in C_{n-1} \text{ or } u \in C_{n-1},\ v \in C\}$$
  Finally, define
  $$C_\infty = C_1 \cup C_2 \cup C_3 \cup \cdots$$
- Theorem: the code C is uniquely decodable if and only if C and C_∞ are disjoint.
- We won't both much with this theorem,
  since as we'll see it isn't of much practical use.

A Check for Instantaneous Codes

- A code is instantaneous if and only if no codeword is a prefix of some other codeword. (ie if C_i is a codeword, $C_i Z$ cannot be a codeword for any Z). This is a *prefix code*.
- Proof:
  ($\Rightarrow$) If codeword $C(a_i)$ is a prefix of codeword $C(a_j)$, then the encoding of the sequence $\mathbf{x} = a_i$ is obviously a prefix of the encoding of the sequence $\mathbf{x}' = a_j$.
  ($\Leftarrow$) If the code is not instantaneous, let $\mathbf{z} = C(\mathbf{x})$ be an encoding that is a prefix of another encoding $\mathbf{z}' = C(\mathbf{x}')$, but with $\mathbf{x}$ not a prefix of $\mathbf{x}'$, and let $\mathbf{x}$ be as short as possible.
  The first symbols of $\mathbf{x}$ and $\mathbf{x}'$ can't be the same, since if they were, we could drop these symbols and get a shorter instance. So these two symbols must be different, but one of their codewords must be a prefix of the other.

Existence of Codes

- Since we hope to compress data, we would like codes that are uniquely decodable and whose codewords are short.
- Also, we'd like to use instantaneous codes where possible since they are easiest and most efficient to decode.
- If we could make all the codewords really short, life would be really easy. Too easy. Why?
  Because there are only a few possible short codewords and we can't reuse them or else our code wouldn't be decodable.
- Instead, making some codewords short will require that other codewords be long, if the code is to be uniquely decodable.
- **Question 1:** What sets of codeword lengths are possible?
- **Question 2:** Can we always manage to use instantaneous codes?

McMillan's Inequality

- There is a uniquely decodable binary code with codewords having lengths $l_1, \ldots, l_I$ if and only if

$$\sum_{i=1}^{I} \frac{1}{2^{l_i}} \leq 1$$

- E.g. there is a uniquely decodable binary code with lengths $1,\ 2,\ 3,\ 3$, since

$$1/2 + 1/4 + 1/8 + 1/8 = 1$$

- An example of such a code is $\{0,\ 01,\ 011,\ 111\}$.
- There is *no* uniquely decodable binary code with lengths $2,\ 2,\ 2,\ 2,\ 2$, since

$$1/4 + 1/4 + 1/4 + 1/4 + 1/4 > 1$$

We Can Always Use Instantaneous Codes

- Since instantaneous codes are a proper subset of uniquely decodable codes, we might have expected that the condition for existence of a u.d. code to be less stringent than that for instantaneous codes.
- But combining Kraft's and McMillan's inequalities, we conclude that there is an instantaneous binary code with lengths $l_1, \ldots, l_I$ if and only if there is a uniquely decodable code with these lengths.
- **Implication:** There is probably no practical benefit to using uniquely decodable codes that aren't instantaneous.
- **Happy consequence:** We don't have to worry about how the encoding is terminated (if at all) or about decoding delays (at least for symbol codes; for block codes this will change).

Kraft's Inequality

- There is an instantaneous binary code with codewords having lengths $l_1, \ldots, l_I$ if and only if

$$\sum_{i=1}^{I} \frac{1}{2^{l_i}} \leq 1$$

- This is exactly the same condition as McMillan's inequality!
- E.g. there is an instantaneous binary code with lengths $1,\ 2,\ 3,\ 3$, since

$$1/2 + 1/4 + 1/8 + 1/8 = 1$$

- An example of such a code is $\{0,\ 10,\ 110,\ 111\}$.
- There is an instantaneous binary code with lengths $2,\ 2,\ 2$, since

$$1/4 + 1/4 + 1/4 < 1$$

- An example of such a code is $\{00,\ 10,\ 01\}$.

Proving the Two Inequalities

- We can prove both Kraft's and McMillan's inequality by proving that for any set of lengths, $l_1, \ldots, l_I$, for binary codewords:

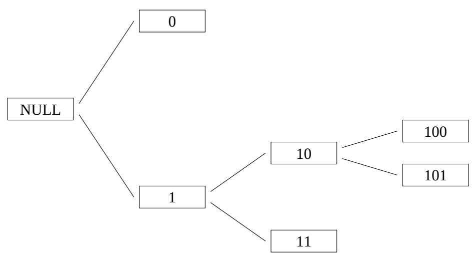
A) If $\sum_{i=1}^{I} 1/2^{l_i} \ \leq\ 1$, we can construct an instantaneous code with codewords having these lengths.

B) If $\sum_{i=1}^{I} 1/2^{l_i} \ >\ 1$, there is no uniquely decodable code with codewords having these lengths.

- (A) is half of Kraft's inequality.
  (B) is half of McMillan's inequality.
- Using the fact that instantaneous codes are uniquely decodable, (A) gives the other half of McMillan's inequality, and (B) gives the other half of Kraft's inequality.
- To do this, we'll introduce a helpful way of thinking about codes as...trees!

Visualizing Prefix Codes as Trees

- We can view codewords of an instantaneous (prefix) code as leaves of a tree.
- The root represents the null string; each level corresponds to adding another code symbol.
- Here is the tree for a code with codewords 0, 11, 100, 101:



Constructing Instantaneous Codes

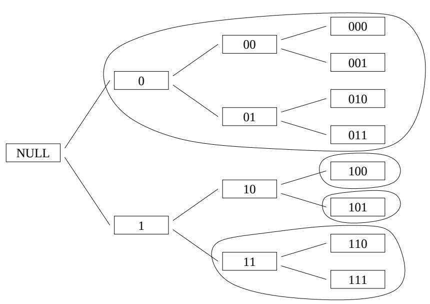
- Suppose that Kraft's Inequality holds:

$$\sum_{i=1}^{I} \frac{1}{2^{l_i}} \leq 1$$

- Order the lengths so $l_1 \leq \cdots \leq l_I$.
- Q: In the binary tree with depth l_I, how can we allocate subtrees to codewords with these lengths?
- A: We go from shortest to longest, $i = 1, \ldots, I$:
  1) Pick a node at depth l_i that isn't in a subtree previously used, and let the code for codeword i be the one at that node.
  2) Mark all nodes in the subtree headed by the node just picked as being used, and not available to be picked later.
- Let's look at an example...

Extending the Tree to Maximum Depth

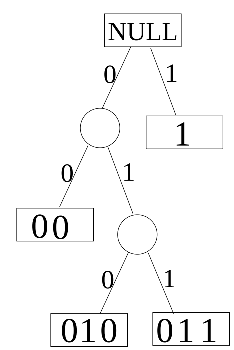
- We can extend the tree by filling in the subtree underneath every actual codeword, down to the depth of the longest codeword.
- Each codeword then corresponds to either a leaf or a subtree.
- Previous tree extended, with each codeword's leaf or subtree circled:



- Short codewords occupy *more* of the tree. For a binary code, the fraction of leaves taken by a codeword of length l is $1/2^l$.

Building an Instantaneous Code

- Let the lengths of the codewords be {1,2,3,3}.
- First check: $2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} \leq 1$.
- Our final code can be read from the leaf nodes: {1,00,010,011}.



Construction Will Always Be Possible

- Q: Will there always be a node available in step (1) above?
- If Kraft's inequality holds, we will always be able to do this.
- To begin, there are 2^{l_b} nodes at depth l_b.
- When we pick a node at depth l_a, the number of nodes that become unavailable at depth l_b (assumed not less than l_a) is $2^{l_b-l_a}$.
- When we need to pick a node at depth l_j, after having picked earlier nodes at depths l_i (with $i < j$ and $l_i \leq l_j$), the number of nodes left to pick from will be

$$2^{l_j} - \sum_{i=1}^{j-1} 2^{l_j-l_i} = 2^{l_j}\left[1 - \sum_{i=1}^{j-1}\frac{1}{2^{l_i}}\right] > 0$$

  Since $\sum\limits_{i=1}^{j-1} 1/2^{l_i} < \sum\limits_{i=1}^{I} 1/2^{l_i} \leq 1$, by assumption.
- This proves (A).

UD Codes Must Obey the Inequality

- Let $l_1 \leq \cdots \leq l_I$ be the codeword lengths. Define $K = \sum_{i=1}^{I} \frac{1}{2^{l_i}}$.
- For any positive integer n, we sum over all possible combinations of values for $i_1, \ldots, i_n$ in $\{1, \ldots, I\}$.

$$K^n = \sum_{i_1,\ldots,i_n} \frac{1}{2^{l_{i_1}}} \times \cdots \times \frac{1}{2^{l_{i_n}}}$$

- We rewrite this in terms of possible values for $j = l_{i_1} + \cdots + l_{i_n}$:

$$K^n = \sum_{j=1}^{nl_I} \frac{N_{j,n}}{2^j}$$

  $N_{j,n}$ is the $\#$ of sequences of n codewords with total length j.
- If the code is uniquely decodable, $N_{j,n} \leq 2^j$, so $K^n \leq nl_I$, which for big enough n is possible only if $K \leq 1$.
- This proves (B). (For instantaneous codes, the intuition is that short codes "use up" their subtree.)