Lecture 3:

Proving the Kraft-McMillan Inequalities

September 18, 2006

Review: Course Content

- Lossless Data Compression
  Shannon's Noiseless Coding Theorem
  Lower limit on lossless compression is the "source entropy".
  Algorithms: Huffman Coding, Arithmetic Coding
- Transmission over Noisy Channels
  Shannon's Noisy Coding Theorem
  Upper limit on error-free transmission rate is "channel capacity".
  Algorithms: Linear Codes, Low Density Parity Check Codes
- (*)Lossy Compression
  Shannon's Rate-Distortion Theorem
  Algorithms: mp3,jpeg,mpeg

Review: Our Approach

- The study of both compression and transmission requires that we abstract data and messages as sequences of symbols from a finite alphabet (ignoring semantics of content).
- Both problems involve two distinct tasks:
  1) *Modeling*. We have to represent the stochastic behaviour of the source and the channel using *probabilistic models*.
  2) *Encoding/Decoding*. Given our source and channel models we want to algorithmically design schemes for compression and transmission that have certain good properties (correct/efficient/optimal).
- For now, we are assuming the models are known and are focusing on the codes. But later in the course we will study adaptive/dictionary methods (e.g. Lempel-Ziff,gzip,PPM) which combine modeling and coding together.

Review: Mathematical Setup

- A stochastic source emits a sequence of symbols (from alphabet $\mathcal{A}$) $X = X_1, X_2, \ldots, X_N$ with probability $p(X)$.
- Our encoder (code) C converts this into an (bitstring) encoding Z.
- We assume (for now) that the decoder can see Z exactly (noiseless channel), that we are required to reconstruct X exactly (lossless compression) and that we are using a *symbol code*, (i.e. we encode each symbol X_i independently and concatenate their encodings).
- We require the code to be uniquely decodable (UD), and we saw that for any UD code there is always an instantaneously decodable (ID) code with the same codeword lengths. These lengths must satisfy the Kraft-McMillan inequality: $\sum_i 2^{-l_i} \leq 1$.
- We will measure the quality of our code by the *average length* (under $p(X)$) of the encoding Z, compared to the length of X.

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:

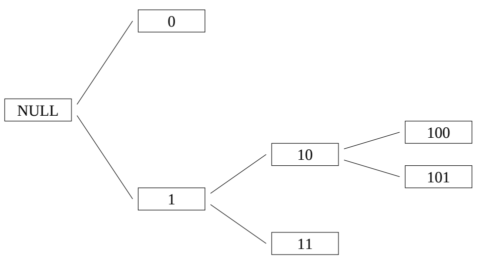
A) If $\sum_{i=1}^{I} 1/2^{l_i} \leq 1$, we can construct an instantaneous code with codewords having these lengths.

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!

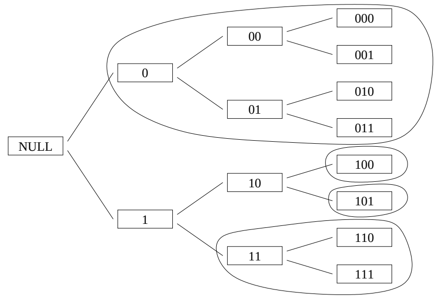
Visualizing Prefix-Free Codes as Trees

- We can view codewords of an instantaneous (prefix-free) 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:



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 $1/2^l$.

Constructing Instantaneous Codes

- Suppose that Kraft's Inequality holds:

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

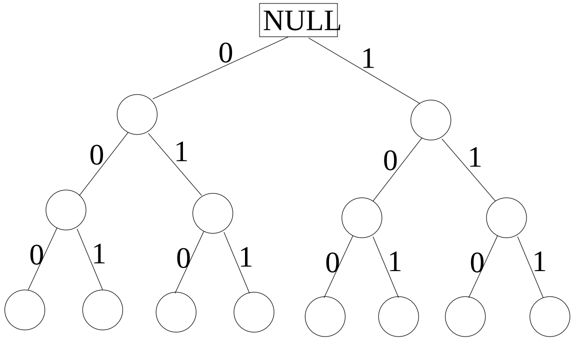
- Order the lengths so $l_1 \leq \cdots \leq l_I$.
- Q: In the binary tree with depth l_I, how can we allocate subtrees to codewords with these lengths?
- A: We go from shortest to longest, $i = 1, \ldots, I$:

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...

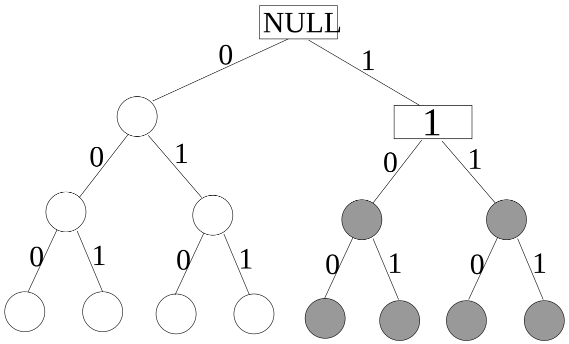
Building an Instantaneous Code (0)

- Let the lengths of the codewords be {1,2,3,3}.
- First check: $2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} \leq 1$.
- Initialize the tree (level 0).



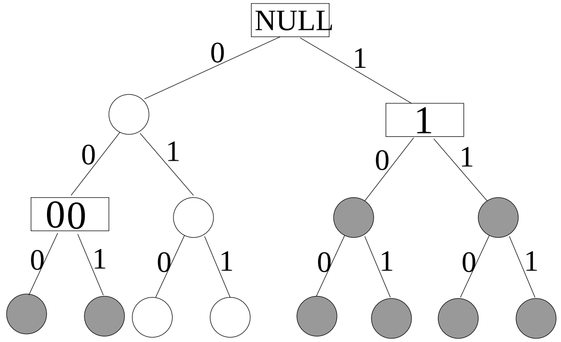
Building an Instantaneous Code (1)

- Let the lengths of the codewords be {1,2,3,3}.
- Pick (arbitrarily) an unmarked node at level 1 to use for codeword of length 1; mark the subtree below it.



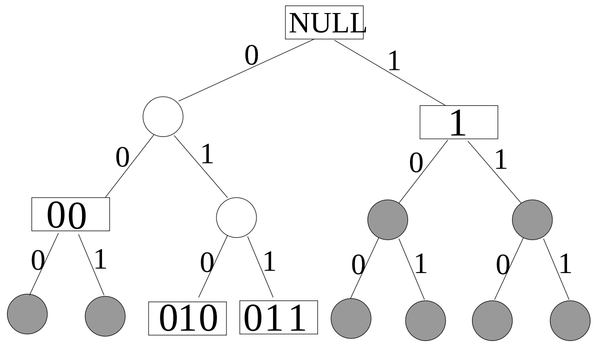
Building an Instantaneous Code (2)

- Let the lengths of the codewords be {1,2,3,3}.
- Pick (arbitrarily) an unmarked node at level 2 to use for codeword of length 2; mark the subtree below it.



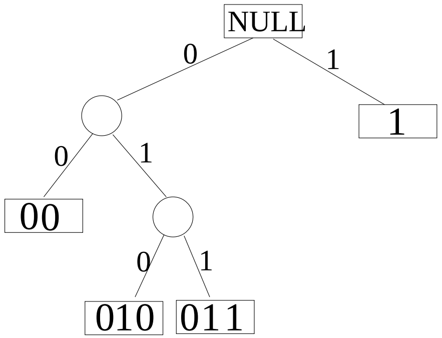
Building an Instantaneous Code (3)

- Let the lengths of the codewords be {1,2,3,3}.
- Pick two unmarked nodes at level 3 as codewords of length 3.



Building an Instantaneous Code 12

- Let the lengths of the codewords be {1,2,3,3}.
- Our final code can be read from the leaf nodes: {1,00,010,011}.



Construction Will Always Be Possible 13

- 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 $l_i \le l_j$), the number of nodes left to pick from will be an integer equal to

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

Since $\sum\limits_{i=1}^{j-1} 1/2^{l_i} \; < \; \sum\limits_{i=1}^{I} 1/2^{l_i} \; \le \; 1$, by assumption.

- This proves we can always construct an ID code if $\sum_i 2^{-l_i} \le 1$.

UD Codes Must Obey the Inequality 14

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

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

- 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, $N_{j,n} \le 2^j$, so $K^n \le nl_I$, which for big enough n is possible only if $K \le 1$.
- This proves that any UD code must satisfy $\sum_i 2^{-l_i} \le 1$.

Tradeoffs Choosing Codeword Lengths 15

- The Kraft-McMillan inequalities imply that to make some codewords shorter, we will have to make others longer.
- Example: The obvious binary encoding for eight symbols uses codewords that are all three bits long. This code is instantaneous, and satisfies the Kraft inequality, since:

$$\frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} = 1$$

- Suppose we want to encode the first symbol using only two bits. We'll have to make some other codewords longer – eg, we can encode two of the other symbols in four bits, and the remaining five symbols in three bits, since

$$\frac{1}{2^2} + \frac{1}{2^4} + \frac{1}{2^4} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} = 1$$

How should we choose among the possible codes?

Formalizing Which Codes are the Best: Probabilities for Source Symbols

- We'd like to choose a code that uses short codewords for common symbols and long ones for rare symbols.
- To formalize this, we need to assign each symbol in the source alphabet a probability.
- Symbols $a_1, \ldots, a_I$ will have probabilities written as $p_1, \ldots, p_I$. We assume that these probabilities don't change with time.
- We also assume that symbols in the source sequence, $X_1, X_2, \ldots, X_N$, are *independent*:

$$\begin{aligned} P(X_1 = a_{i_1},\ X_2 = a_{i_2},\ \ldots,\ X_n = a_{i_N}) \\ = \prod_n P(X_n = a_{i_n}) = p_{i_1} p_{i_2} \cdots p_{i_N} \end{aligned}$$

- These assumptions are really too restrictive in practice, but we'll ignore that for now.

Expected Codeword Length

- Consider a code whose codewords for symbols $a_1, \ldots, a_I$ have lengths $l_1, \ldots, l_I$. Let the probabilities of these symbols be $p_1, \ldots, p_I$. We define the *expected codeword length* for this code to be

$$L \;=\; L(C, X) = \sum_{i=1}^{I} p_i l_i$$

- This is the average length of the codeword encoding a single source symbol. But since averaging is a linear operation, the average length of a coded message with N source symbols is just NL.
- We aim to choose a code for which L is small.
- Basically, we want to assign short codeword lengths to the more probable symbols but we also need to satisfy the KM inequality so we will be forced to assign longer lengths to the less probable symbols.

Optimal Codes

- We say a code is *optimal* for a given source (with given symbol probabilities) if its average length is at least as small as that of any other code. (There can be many optimal codes for the same source, all with the same average length.)
- The Kraft-McMillan inequalities imply that if there is an optimal code, there is also an optimal *instantaneous* code. More generally, for any uniquely decodable code with average length L, there is an instantaneous code with the same average length.
- Questions: Can we figure out the codeword lengths of an optimal code starting from the symbol *probabilities*? i.e. can we solve:

$$\min_{\{l_i\}} \sum_i p_i l_i \qquad \text{subject to} \sum_i 2^{-l_i} \leq 1$$

  Can we find such an optimal code, and use it in practice?
- Answers: next class!