

Teaching Lossless Data Compression

Sergio Verdú

Most courses on information theory, particularly those patterned after Cover and Thomas [1], cover the algorithmic side of lossless data compression, most notably Huffman, arithmetic and Lempel-Ziv codes. I like to do it at the advanced-undergraduate level and it is great fun to teach. However, how we describe and analyze those algorithms is not the purpose of this column. Instead, expanding on one of the items in my Shannon Lecture, I will discuss the teaching of the fundamental limits of lossless data compression.

Although there are other source coding setups, such as (Tunstall) variable-to-fixed coding, the conventional canon found in most textbooks, and taught in most information theory courses, deals with two separate source coding paradigms:

- 1) Variable-length symbol-by-symbol lossless compression;
- 2) Fixed-length (or fixed-to-fixed) almost-lossless compression.

The centerpieces of the fundamental limits of symbol-by-symbol lossless compression are a converse result that states that the code-lengths of any uniquely decodable code must satisfy the Kraft inequality, and an achievability result that guarantees the existence of a prefix code with any set of code-lengths that satisfies the Kraft inequality. Therefore, non-prefix codes do not offer the prospect of increased efficiency. The converse result readily leads to the conclusion that the average length of a variable-length symbol-by-symbol uniquely decodable code cannot be smaller than the source entropy, a result commonly, and wrongly, attributed to Shannon. A formula for the minimal average length is not known, but an algorithm to compute it is indeed known (provided the alphabet is finite) since it is achieved by the Huffman code. Moreover, since the achievability result guarantees that a code exists whose code-lengths are equal to the log-reciprocal-probabilities (aka “ideal code-lengths”) rounded-up, we can upper bound the minimal average length of a binary symbol-by-symbol lossless code by the entropy plus one bit.

In contrast to the analysis of variable-length codes, the conventional analysis of fixed-length almost-lossless compression is asymptotic in nature and does not just deal with averages: instead, by invoking Shannon’s notion of typicality and the law of large numbers, it shows that as long as the coding rate exceeds the entropy and the source is memoryless, vanishing error probability is achievable. It is also a good idea for the teacher to show the converse of this statement. See, for example, the proof of [2, Theorem 1.1.1].

Some natural questions that the student may ask:

- Arithmetic codes and Lempel-Ziv codes are strictly lossless and have variable length, but they are not symbol-by-symbol codes. Why did we restrict the analysis of variable-length codes to symbol-by-symbol codes?
- Are fixed-length almost-lossless codes used in practice?
- Why did we limit our analysis of the fundamental limits to memoryless sources? Aren’t sources in the real world redundant because of their memory?
- If symbol-by-symbol codes cannot exploit memory, in what applications are they used?
- Attaining good average length at the expense of large variance, or large probability of exceeding a given threshold, may not be desirable. For variable-length lossless codes, is there anything beyond the minimal average length that we can analyze?
- There are about 2^{922} possible tweets (twitter messages are limited to no more than 140 characters). How many bits are required to compress 95% of the time?

First, let us address the issue of memory. Yes, if we do not exploit the memory in discrete sources such as text and digital photographs we do a really lousy job at compressing. But compression of redundant memoryless sources is much more common than one may think, primarily because through various reversible transformations such as run-length encoding (e.g. fax), linear transforms (e.g. JPEG) and the Burrows-Wheeler transform (e.g. bzip) the redundancy in memory is shifted to non-equiprobability of the first-order distribution. In those settings, symbol-by-symbol codes, and in particular Huffman codes find plenty of applications. But as far as the fundamental limits, I feel remiss if I end my coverage with memoryless sources. After all, Shannon gave the the asymptotic equipartition property [3, Theorem 3] not just for memoryless sources but for Markov sources. Pedagogically, after treating the memoryless case, it is not too difficult to prove the asymptotic equipartition property for stationary ergodic sources using Markov approximation: either by McMillan’s original approach (see Gallager [4]) or the Algoet-Cover sandwich method [1].

Next, let us revisit the traditional restriction of the analysis of strictly lossless codes to symbol-by-symbol codes. It is not as severe as it sounds because we can always think of super-symbols, each encompassing m consecutive symbols of the original source. Then, the minimal average length is the entropy of m -words plus at most 1. This trick is more useful conceptually than algorithmically, since the computational complexity skyrockets with m . Algorithmically, a preferable way to deal with memory is to abandon the symbol-by-symbol paradigm altogether and use arithmetic coding or Lempel-Ziv codes. As far as the analysis of the ultimate efficiency, we can go one step further and dispense with super-symbols altogether by viewing the whole file to be compressed as one symbol. But here’s the thing: we can get better efficiency than what the conventional analysis taught in textbooks predicts! To fix ideas, consider the twitter question. Suppose we knew the probabilities of all possible tweets; what would be the best lossless data compressor? Would it be a Huffman code for a source with an alphabet of 2^{922} “symbols”? The average length of the Huffman-coded version will be equal to the entropy of the tweet distribution plus at

most 1 bit. But we can do better: list all the tweets in decreasing probabilities and encode them with the binary strings:

$$\{\emptyset, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, \dots\}$$

This is the best code not just in the sense of minimizing the average, but in the much stronger sense of maximizing the cdf of the length at every point. Therefore, the abscissa at which the cdf of the length reaches 0.95 is the answer to the foregoing question. How much better than the Huffman code is the optimal variable-length code? Its average length is in fact *upper* bounded by the entropy (the ideal code-lengths are not so ideal after all) and admits the expression

$$\sum_{k=1}^{\infty} \mathbb{P}[X \geq 2^k]$$

where the source realizations have been relabeled so that the integer $X = \ell$ is the ℓ th most probable outcome. According to recent results [5], for memoryless sources of length n and entropy H , the minimal average length of a fixed-to-variable code behaves as $nH - (1/2)\log n + O(1)$, rather than $nH + O(1)$ for a super-symbol Huffman code or an arithmetic code. (The symbol-by-symbol Huffman code would be even worse: $n(H + g)$ for $0 \leq g < 1$.) Wait. Wasn't the entropy a sacrosanct limit that no variable-length code could beat? That is true for symbol-by-symbol uniquely decodable codes; however, once the super-symbol becomes the whole file to compress we should realize that the prefix condition (or the uniquely decodable condition) is superfluous. But then, how do we know where the compressed file ends? Just think of a file stored in a hard disk. Does it satisfy the prefix condition (in some humongous tree) so it can tell us by itself where its "ends"? No. In fact, it will probably be stored in many chunks identified by a directory of pointers. Admittedly, the optimal non-prefix variable-length code is easier said than done unless the source model is very simple; moreover, if the file to compress is long enough, then the penalty for imposing the extraneous prefix condition on the encoded file gets amortized. Still, in the short run and in particular if we also pay attention to variance in addition to average, there are efficiencies to reap.

And finally, let us address why we study almost-lossless n -to- k codes. In those codes, one of the binary output k -strings signals that the input realization cannot be handled by the compressor. Although all practical data compressors are strictly lossless and therefore variable-length, we can easily turn an almost-lossless code into a lossless code by, for example, substituting the special output k -string by the input string. Conversely, we can easily turn a variable-length code into an almost-lossless fixed-length code. Another important motivation for teaching

them is that not only do almost-lossless source codes provide the reason to study the asymptotic equipartition property, but they are the indispensable gateway to many other results in information theory, such as the source-channel separation theorem, the Slepian-Wolf theorem and the random binning proof method, ubiquitous in multiuser information theory. Furthermore, linear channel coding is the dual problem to linear fixed-to-fixed source coding; in fact, excellent data compressors can be built based on modern sparse-graph codes [6]. These are all good motivations to teach almost lossless compression, but in fact I would argue that the main reason is a very simple fact (proof left to the reader) that has gone unrecognized because of the traditional fixation with Kraft-inequality compliant variable-length codes: Regardless of the source, the minimal probability of error of an n -to- k code is equal to the probability that the length of the optimal variable-length code for X_1, \dots, X_n is greater than or equal to k . Therefore, the analysis (asymptotic or not) of the fundamental limits of fixed-to-fixed data compression is equivalent to the analysis of the fundamental limits of fixed-to-variable data compression.

Sergio Verdú is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544, USA, Email: verdu@princeton.edu

References

- [1] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. New York: Wiley, 2006.
- [2] I. Csiszár and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems*. New York: Academic, 1981.
- [3] C. E. Shannon, "A Mathematical Theory of Communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, Jul.–Oct. 1948.
- [4] R. G. Gallager, *Information Theory and Reliable Communication*. New York: Wiley, 1968.
- [5] W. Szpankowski and S. Verdú, "Minimum Expected Length of Fixed-to-Variable Lossless Compression without Prefix Constraints," *IEEE Trans. on Information Theory*, to appear.
- [6] G. Caire, S. Shamai, A. Shokrollahi and S. Verdú, "Fountain Codes for Lossless Data Compression," *Algebraic Coding Theory and Information Theory*, A. Ashikhmin, A. Barg, I. Duursma, Eds., DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 68, pp. 1–20, American Mathematical Society, 2006.