Compressing combinatorial objects

Christian Steinruecken
University of Cambridge

Abstract

Most of the world's digital data is currently encoded in a sequential form, and compression methods for sequences have been studied extensively. However, there are many types of non-sequential data for which good compression techniques are still largely unexplored. This paper contributes insights and concrete techniques for compressing various kinds of non-sequential data via arithmetic coding, and derives re-usable probabilistic data models from fairly generic structural assumptions. Near-optimal compression methods are described for certain types of permutations, combinations and multisets; and the conditions for optimality are made explicit for each method.

1 Introduction

Much of data compression is concerned with the task of compressing files that commonly occur in every day usage. Improvements in compression effectiveness inevitably stem from improvements in data modelling, and the strength of any given model clearly depends on the type of data that is being compressed. The need for good compression thus motivates developing models that are good at predicting the data of interest. The development of appropriate data models can be difficult, especially when computational constraints are taken into account. For example, models that are good at compressing DNA sequences are different from models that are good at compressing human text. Modern file compressors often include many different data models (often for specific types of data) whose predictions are combined [1, 2, 3].

Sequence compression algorithms have been studied extensively, not least because sequences are the most common form in which data is represented in a computer. Of course, a serial representation is not always a natural description of data (e.g. 2-dimensional pictures, elements of a dictionary, the hierarchical structure of a file system), and is often chosen somewhat arbitrarily.

This paper explores data models (and compression algorithms) for some basic types of data that are non-sequential, or have particular symmetries imposed on them. In particular, various models are proposed for compressing different kinds of permutations, combinations, and multisets.

The methods for compressing data with these structures can be used as components in larger models and compression algorithms; we also hope that the insights presented for the objects here may be useful for finding algorithms for other kinds of combinatorial objects.

The rest of this paper is structured as follows. *Section 2* reviews the compression of sequences whose elements are independent and identically distributed from a known distribution. *Section 3* introduces multisets and describes an optimal compression scheme for multisets whose elements are distributed the same way as in section 2. *Section 4* introduces permutations and describes an optimal compression scheme for permutations that are uniformly distributed. *Section 5* shows that a sequence can be split into a multiset and a permutation, and that the information content of these two components sum to the same value as the information content of the original sequence. *Section 6* extends the coding method for permutations to truncated permutations. *Section 7* introduces combinations and a compression scheme for them. *Section 8* discusses uniformly distributed multisets and describes a compression scheme for them. Finally, *Section 9* extends the results of previous sections to the case that the symbol distribution is unknown and needs to be learned adaptively.

2 Sequences

We'll start by reviewing a task that is well known: the compression of a fixed-length sequence $x_1 \ldots x_N$ of elements from a finite alphabet $\mathcal{X}$. Rather than discussing a specialised model that is good at compressing text or other real-world sequences, we'll look at an oversimplified sequence model and then show equivalents for non-sequential forms of data. So let's assume for now that each element x_n in the sequence is independently and identically distributed (*iid*) according to some distribution D. The probability of such a sequence is:

$$\Pr(x_1 \ldots x_N \mid D, N) = \prod_{n=1}^{N} D(x_n). \tag{1}$$

Sequences of this kind can be compressed with an arithmetic coding scheme [4, 5], where the symbols x_n are encoded sequentially using D at each coding step.[1] This kind of algorithm produces compressed output whose total length is guaranteed to be within 2 bits of the theoretic optimum – the *Shannon information content* of the sequence under the above assumptions:

$$\log_2 \frac{1}{\prod_{n=1}^{N} D(x_n)} \;=\; \sum_{n=1}^{N} \log_2 \frac{1}{D(x_n)}. \tag{2}$$

This guarantee is strictly stronger than proving the *expected* output length to be within 2 bits of the Shannon entropy of D, because the guarantee holds for every input sequence, not just on average. Algorithms of the above kind are well known, and form the basis of many sequence compression algorithms.[2]

This paper now explores similarly fundamental compression models for input objects that *aren't* sequences. Compression of non-sequential data was considered by

[1]There are some fairly minor constraints, e.g. that it must be possible to discretise D to the resolution of the arithmetic coder.

[2]Of course, there are many ways of modelling sequences, and the above simplistic model, while fundamental, is hardly suitable for compressing real-life files. In fact, if D were chosen to be a uniform distribution, the above method achieves no compression at all, as all sequences of length N are assumed to have equal probability.

Varshney and Goyal [6, 7, 8], and this paper follows up on their work. The models in this paper have been chosen primarily for simplicity and clarity of presentation; more elaborate models for combinatorial objects can be found e.g. in [9].

3 Multisets

A *multiset* $\mathcal{M}$ is an unordered collection of elements, where elements may occur multiple times. Let $\mathcal{M}(x)$ denote how often element x occurs in $\mathcal{M}$, and $|\mathcal{M}|$ the total number of elements in $\mathcal{M}$ (including repetitions). For example, $\mathcal{K} = \{\mathtt{A}, \mathtt{B}, \mathtt{B}\}$ is a multiset with $|\mathcal{K}| = 3$, $\mathcal{K}(\mathtt{A}) = 1$, and $\mathcal{K}(\mathtt{B}) = 2$.

One can think of a multiset as the *histogram* that captures the symbol occurrence counts in a sequence $x_1 \ldots x_N$, such that:

$$|\mathcal{M}| \;=\; \sum_{x \in \mathcal{X}} \mathcal{M}(x) \;=\; N. \tag{3}$$

A multiset represents the information that remains when the order information of a sequence is lost.

Analogously to how we considered a sequence of N draws in the previous section, consider now a multiset $\mathcal{M}$ that was created by making N independent draws from a known distribution D over the same finite alphabet $\mathcal{X}$. The probability of such a multiset $\mathcal{M}$ is:

$$\Pr(\mathcal{M} \mid N, D) = N! \cdot \prod_{x \in \mathcal{X}} \frac{D(x)^{\mathcal{M}(x)}}{\mathcal{M}(x)!}. \tag{4}$$

As $\mathcal{M}$ is a multivariate object, the above distribution cannot be interfaced directly to an arithmetic coder; arithmetic coders can typically only encode (sequences of) discrete univariate choices. One possible solution is to factorise (4) into a product of univariate distributions as follows:

$$\Pr(\mathcal{M} \mid N, D) = \prod_{x \in \mathcal{X}}^{(<)} \text{Binomial}\left(\mathcal{M}(x) \,\middle|\, |\mathcal{M}| - \sum_{y<x} \mathcal{M}(y),\; \frac{D(x)}{1 - \sum_{y<x} D(y)} \right) \tag{5}$$

where $\text{Binomial}(k \mid K, \theta) = \binom{K}{k} \theta^k (1-\theta)^{K-k}$ is the binomial distribution, $<$ is a total ordering relation of the alphabet $\mathcal{X}$, and the product iterates over the elements of $\mathcal{X}$ in the order defined by $<$. A derivation of this factorisation can be found in [9].

$\mathcal{M}$ can then be compressed with an arithmetic coding scheme that sequentially encodes each of the components $\mathcal{M}(x)$ using the conditional distributions given in (5). The length of the resulting compressed output sequence is within 2 bits of $\mathcal{M}$'s information content, i.e. the optimal length in bits:

$$\log_2 \frac{1}{\Pr(\mathcal{M} \mid N, D)} \;=\; \log_2 \frac{1}{N!} + \sum_{x \in \mathcal{X}} \mathcal{M}(x) \log_2 \frac{1}{D(x)} + \sum_{x \in \mathcal{X}} \log_2 \mathcal{M}(x)! \tag{6}$$

4 Permutations

A *permutation* is an arrangement of elements in a given multiset. For example, the multiset $\{\mathtt{X}, \mathtt{Y}, \mathtt{Z}\}$ has 6 possible permutations:

$$(\mathtt{X},\mathtt{Y},\mathtt{Z}) \qquad (\mathtt{X},\mathtt{Z},\mathtt{Y}) \qquad (\mathtt{Y},\mathtt{X},\mathtt{Z}) \qquad (\mathtt{Y},\mathtt{Z},\mathtt{X}) \qquad (\mathtt{Z},\mathtt{X},\mathtt{Y}) \qquad (\mathtt{Z},\mathtt{Y},\mathtt{X}),$$

and the multiset $\{\blacktriangledown, \blacktriangle, \blacktriangle\}$ has three permutations: $(\blacktriangledown, \blacktriangle, \blacktriangle)$, $(\blacktriangle, \blacktriangledown, \blacktriangle)$ and $(\blacktriangle, \blacktriangle, \blacktriangledown)$. Despite what the above notation might suggest, a permutation contains no information about symbols or their occurrence counts: it merely specifies an arrangement of symbols for a *given* multiset. A sequence can always be separated into a multiset and a permutation of that multiset.
The number of possible permutations of a given multiset $\mathcal{M}$ is exactly

$$\frac{|\mathcal{M}|!}{\prod_{x\in\mathcal{X}} \mathcal{M}(x)!}, \tag{7}$$

so a uniform distribution over permutations assigns to each permutation the probability:

$$\Pr\big(x_1 \,...\, x_{|\mathcal{M}|} \,\big|\, \mathcal{M}\big) = \frac{1}{|\mathcal{M}|!} \cdot \prod_{x\in\mathcal{X}} \mathcal{M}(x)!. \tag{8}$$

This distribution (8) can be factorised into a product of univariate distributions:

$$\begin{aligned} \Pr(x_1 \,...\, x_N \mid \mathcal{M}) &= \prod_{n=1}^{N} \Pr(x_n \mid \mathcal{M},\ x_1 \,...\, x_{n-1}) && (9) \\ &= \prod_{n=1}^{N} \frac{\mathcal{M}(x_n) - \boxed{\sum_{k=1}^{n-1} \mathbb{1}[x_k = x_n]}}{N-n+1}, && (10) \end{aligned}$$

where the shaded box counts the number of times symbol x_n occurs in the preceding sequence $x_1 \,...\, x_{n-1}$ ($\mathbb{1}[\,\mathrm{Q}\,] = 1$ if the expression Q is true, and 0 otherwise). One way to store such a permutation with an arithmetic coder is to encode each x_n with the conditional distribution (10), iterating over the indices n of the permutation. This procedure produces an output length that is within 2 bits of the Shannon information content of the permutation:

$$\log_2 \frac{1}{\Pr\big(x_1 \,...\, x_{|\mathcal{M}|} \,\big|\, \mathcal{M}\big)} \quad = \quad \log_2 |\mathcal{M}|! - \sum_{x\in\mathcal{X}} \log_2 \mathcal{M}(x)! \tag{11}$$

5 Sequence = Multiset + Permutation

Any sequence $x_1 \,...\, x_N$ can be split into a multiset $\mathcal{M}$ and a permutation of $\mathcal{M}$. For the case that each element x_n is independent and identically distributed according to some known distribution D, we gave coding schemes for each of these three objects. These coding schemes are optimal in the sense that they compress each object to within 2 bits of the object's Shannon information content.

The information content of the multiset was given in (6), and of the multiset's permutation in (11). Recalling that each symbol $x \in \mathcal{X}$ occurs $\mathcal{M}(x)$ times in the sequence x_1 ... x_N, and that $N = |\mathcal{M}|$, the sum of quantities (6) and (11) equals:

$$\sum_{x \in \mathcal{X}} \mathcal{M}(x) \log_2 \frac{1}{D(x)} \quad = \quad \sum_{n=1}^{N} \log_2 \frac{1}{D(x_n)}. \tag{12}$$

This quantity is the same as expression (2), the information content of the sequence.

The above coding methods were derived for sequences and multisets that made the oversimplifying assumption that each element x_n in the sequence was drawn independently from a known distribution D. In reality, distributions over elements are not typically known in advance, and should be inferred from the data (both by the compressor and by the decompressor). Section 9 shows how sequences and multisets can be optimally encoded while simultaneously learning an unknown distribution.

Before then, we shall briefly look at truncated permutations, and at two alternative models for multisets.

6 Truncated permutations

The encoding for permutations introduced in section 4 reflects a generative process where elements are drawn without replacement from a given multiset $\mathcal{M}$ until there are no more elements. In this section, we generalise this method to the case that we stop after K draws, resulting in a *truncated permutation* x_1 ... x_K where $K \leq |\mathcal{M}|$.

The probability of drawing a given K-permutation from $\mathcal{M}$ is:

$$\Pr(x_1 \dots x_K \mid K, \mathcal{M}) = \frac{(|\mathcal{M}| - |\mathcal{K}|)!}{|\mathcal{M}|!} \cdot \prod_{x \in \mathcal{M}} \frac{\mathcal{M}(x)!}{(\mathcal{M}(x) - \mathcal{K}(x))!} \tag{13}$$

where $\mathcal{K} \subseteq \mathcal{M}$ summarises the symbol occurrence counts in the truncated permutation x_1 ... x_K. To encode a K-permutation of this kind, the encoding procedure from section 4 can be used, except for stopping after K elements. It might be helpful to note that this early stopping procedure results in a distribution over truncated permutations that is non-uniform, except when $K = |\mathcal{M}|$ or when $\mathcal{M}$ contains no duplicates.

7 Combinations (or submultisets)

In section 3 we considered multisets whose elements were drawn from a distribution D. Rather than drawing elements from a known distribution D, consider drawing elements from a known multiset $\mathcal{M}$ without replacement. If we keep the order of the elements, we end up with a permutation, as discussed in section 4. If the order is thrown away, the resulting structure is called a *combination*.

Suppose we draw K elements from an existing multiset $\mathcal{M}$, where $K \leq |\mathcal{M}|$. The resulting combination $\mathcal{K}$ is a submultiset of $\mathcal{M}$, and has probability:

$$\Pr(\mathcal{K} \mid \mathcal{M}) = \binom{|\mathcal{M}|}{|\mathcal{K}|}^{-1} \cdot \prod_{x \in \mathcal{K}} \binom{\mathcal{M}(x)}{\mathcal{K}(x)} \tag{14}$$

This multivariate distribution can be broken down into a product of univariate distributions, to make it easier to use with an arithmetic coder. For example, (14) can be factorised recursively, using the identity:

$$\Pr(\mathcal{K} \mid \mathcal{M}) \;=\; \Pr(\mathcal{K}(x) \mid \mathcal{M}, |\mathcal{K}|) \cdot \Pr\left(\mathcal{K}_{\setminus\{x\}} \mid \mathcal{M}_{\setminus\{x\}}\right) \tag{15}$$

where x is an arbitrary element that occurs in $\mathcal{K}$, and $\mathcal{K}_{\setminus\{x\}}$ denotes the multiset $\mathcal{K}$ with all occurrences of x removed. The distribution over $\mathcal{K}(x)$ given $\mathcal{M}$ and $|\mathcal{K}|$ is:

$$\Pr(\mathcal{K}(x) \mid \mathcal{M}, |\mathcal{K}|) \;=\; \binom{|\mathcal{M}|}{|\mathcal{K}|}^{-1} \binom{|\mathcal{M}|-\mathcal{M}(x)}{|\mathcal{K}|-\mathcal{K}(x)} \binom{\mathcal{M}(x)}{\mathcal{K}(x)}. \tag{16}$$

8 Uniform multisets

The compression method for multisets described in section 3 assumes that the elements in the multiset are distributed according to some distribution D. This section shows how to compress multisets that are chosen uniformly at random from all possible multisets of a given size N. There are $C_K(N)$ ways of creating an N-size multiset from a set of K distinct elements, where:

$$C_K(N) = \binom{N+K-1}{K-1} = \frac{(N+K-1)!}{N!\;(K-1)!} \tag{17}$$

A uniform distribution over multisets $\mathcal{M}$ (given N and K) therefore assigns each multiset probability mass $\Pr(\mathcal{M} \mid N, K) = 1/C_K(N)$.

9 Sequences and multisets from unknown distributions

Suppose the elements of a sequence $x_1 \ldots x_N$ are independent and identically distributed from some *unknown* distribution D. It is possible to compress sequences of this kind by taking advantage of the knowledge that the sequence's elements are identically distributed: even though the distribution D is not available to the encoder or the decoder, each symbol in the sequence reveals information about D, allowing both encoder and decoder to approximate it.

A simple and well-known technique is to let both encoder and decoder maintain a histogram of symbol occurrence counts, which is updated every time after a symbol in the sequence is encoded or decoded. Each symbol is arithmetically encoded using the best approximation of D, given all the past symbol occurrences and a prior distribution over symbols. Here, this prior is implemented by giving each symbol x an initial pseudocount of $\alpha_x \in \mathbb{R}^+$. The Nth symbol in the sequence can be compressed with an arithmetic coder using the following conditional distribution:

$$\Pr(x_N{=}x \mid x_1 \ldots x_{N-1}, \boldsymbol{\alpha}) \;=\; \frac{\alpha_x + \boxed{\sum_{n=1}^{N-1} \mathbb{1}[x_n = x]}}{A + N - 1} \tag{18}$$

where α_x is the initial count for symbol x, and A is the sum of all α_x. The shaded box counts how often symbol x occurs in the preceding sequence.

The probability distribution over the entire sequence induced by this adaptive scheme can be written as follows:

$$\begin{aligned}\Pr(x_1 \,...\, x_N \mid N, \boldsymbol{\alpha}) &= \prod_{n=1}^{N} \Pr(x_n \mid x_1 \,...\, x_{n-1}, \boldsymbol{\alpha}) && (19)\\ &= \prod_{n=1}^{N} \frac{\alpha_{x_n} + \sum_{k=1}^{n-1} \mathbb{1}[x_k = x_n]}{A + n - 1} && (20)\\ &= \frac{\Gamma(A)}{\Gamma(A+N)} \prod_{x\in\mathcal{X}} \frac{\Gamma(\alpha_x + \mathcal{M}(x))}{\Gamma(\alpha_x)} && (21)\end{aligned}$$

where $\mathcal{M}$ is the symbol histogram of the entire sequence, i.e. $\mathcal{M}(x) = \sum_{n=1}^{N} \mathbb{1}[x_n = x]$, and $\Gamma(\cdot)$ denotes the Gamma function.

We can consider the equivalent scenario for a multiset by throwing away the ordering produced by the above process. As was shown in section 4, a sequence with histogram $\mathcal{M}$ has exactly $|\mathcal{M}|! \cdot \left(\prod_{x\in\mathcal{X}} \mathcal{M}(x)!\right)^{-1}$ permutations. Noting that $|\mathcal{M}| = N$, the distribution (21) over sequences can be converted to a distribution over multisets by dividing out the permutation information:

$$\Pr(\mathcal{M} \mid N, \boldsymbol{\alpha}) = \frac{N!\;\Gamma(A)}{\Gamma(A+N)} \prod_{x\in\mathcal{X}} \frac{\Gamma(\mathcal{M}(x) + \alpha_x)}{\mathcal{M}(x)!\;\Gamma(\alpha_x)} \tag{22}$$

The result is a compound Dirichlet-multinomial distribution, where $\boldsymbol{\alpha}$ are the parameters of the Dirichlet prior. The same result can be obtained by multiplying distribution (4) with a Dirichlet prior, and integrating out D.

To make this distribution suitable for compression with an arithmetic coder, it can be factorised into a product of conditional Beta-binomial distributions, e.g. as follows:

$$\Pr(\mathcal{M} \mid N, \boldsymbol{\alpha}) = \prod_{x\in\mathcal{X}}^{(<)} \mathrm{BetaBin}\left(\mathcal{M}(x) \;\middle|\; N - \sum_{y<x} \mathcal{M}(y),\ \alpha_x,\ A - \sum_{y<x} \alpha_y\right) \tag{23}$$

where the Beta-binomial distribution is univariate, and defined as:

$$\begin{aligned}\mathrm{BetaBin}(k \mid K, \alpha, \beta) &= \int \mathrm{Binomial}(k \mid K, \theta) \cdot \mathrm{Beta}(\theta \mid \alpha, \beta)\; d\theta && (24)\\ &= \binom{K}{k} \cdot \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\;\Gamma(\beta)} \cdot \frac{\Gamma(\alpha+k)\;\Gamma(\beta+K-k)}{\Gamma(\alpha+\beta+K)} && (25)\end{aligned}$$

Using (23) with an arithmetic coder, one can near-optimally encode multisets whose elements are distributed according to an unknown distribution. It's somewhat surprising that it's possible to store a collection of data points in an unordered way, saving the bits that would otherwise be wasted on the ordering. It's even more surprising that this is possible while simultaneously learning the empirical distribution from the data, and exploiting it to compress more effectively. Perhaps one could argue that this kind of compression method is a step towards 'universal' compression of unordered data.

10 Discussion

This paper presented several fundamental models for non-sequential data, and showed how they relate to some known fundamental models for sequences. The approach taken for finding these models was to derive the probability distribution over the combinatorial objects defined by fairly basic assumptions. Where necessary, multivariate distributions were factorised into products of univariate distributions, to make them usable with existing arithmetic coding algorithms.

To encode symbols with an arithmetic coder, one must have a way to compute the cumulative discretised mass of any given symbol under the chosen distribution. In many cases (including that of the Beta-binomial distribution) a "budget allocation algorithm" can be used for this purpose. More efficient methods exist for some distributions, e.g. when the cumulative mass function can be computed in closed form.

The models presented in this paper are not intended to be used for compressing real-world data; they were carefully chosen to be simplistic and flexible, in the hope that they might serve as fundamental components in more elaborate compression algorithms.

There are many other possibilities of defining probability distributions over multisets, and there are many other combinatorial objects worthy of study that were not mentioned in this paper. Plenty of additional models, algorithms and source code related to compression of non-sequential data can be found in [9].

Acknowledgments. The author would like the thank the anonymous reviewers for helpful feedback.

References

[1] M. V. Mahoney, "Fast text compression with neural networks," in *Proceedings of the Thirteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2000)*, J. N. Etheredge and B. Z. Manaris, Eds. AAAI Press, 2000, pp. 230–234.

[2] ——, "The PAQ1 data compression program," 2002, unpublished draft. [Online]. Available: http://cs.fit.edu/~mmahoney/compression/paq1.pdf

[3] ——, "Adaptive weighing of context models for lossless data compression," Department of Computer Science, Florida Institute of Technology, Melbourne, FL, USA., Tech. Rep. CS-2005-16, 2005.

[4] I. H. Witten, R. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, Jun. 1987.

[5] A. Moffat, R. M. Neal, and I. H. Witten, "Arithmetic coding revisited," *ACM Transactions on Information Systems*, vol. 16, no. 3, pp. 256–294, Jul. 1998.

[6] L. R. Varshney and V. K. Goyal, "Ordered and disordered source coding," in *Proceedings of the Information Theory & Applications Inaugural Workshop*, Feb. 2006.

[7] ——, "Toward a source coding theory for sets," in *Proceedings of the Data Compression Conference*, J. A. Storer and M. Cohn, Eds. IEEE Computer Society, 2006, pp. 13–22.

[8] ——, "On universal coding of unordered data," in *Information Theory and Applications Workshop 2007, Conference Proceedings*. IEEE, 2007, pp. 183–187.

[9] C. Steinruecken, "Compressing structured objects," Chapter 5 in *Lossless Data Compression*, Ph.D. dissertation, University of Cambridge, pp. 79–102, 2014.