Sparse Sequential Dirichlet Coding

Joel Veness[†] Marcus Hutter[‡]

[†]University of Alberta, Edmonton, Canada
[‡]Australian National University, Canberra, Australia

June 19, 2012

Abstract

This short paper describes a simple coding technique, Sparse Sequential Dirichlet Coding, for multi-alphabet memoryless sources. It is appropriate in situations where only a small, unknown subset of the possible alphabet symbols can be expected to occur in any particular data sequence. We provide a competitive analysis which shows that the performance of Sparse Sequential Dirichlet Coding will be close to that of a Sequential Dirichlet Coder that knows in advance the exact subset of occurring alphabet symbols. Empirically we show that our technique can perform similarly to the more computationally demanding Sequential Sub-Alphabet Estimator, while using less computational resources.

1 Introduction

Suppose we needed to code a sequence of symbols $x_{1:n} := x_1 x_2 \dots x_n$ from an unknown alphabet $\mathcal{A}$ generated by an unknown memoryless data generating source μ. If we knew an alphabet $\mathcal{X}$ such that $\mathcal{A} \subseteq \mathcal{X}$, one solution would be to code the sequence using the Sequential Dirichlet Estimator

$$\rho_{\mathcal{X}}(x_{1:n}) := \prod_{i=1}^{n} \frac{c(x_{1:i}) + \frac{1}{2}}{i + \frac{|\mathcal{X}|}{2} - 1}, \tag{1}$$

where $c(x_{1:n}) := \sum_{i=1}^{n-1} \mathbb{I}[x_n = x_i]$, as suggested by Krichevsky and Trofimov [1981]. This technique has the property [Tjalkens et al., 1993] that

$$-\log_2 \frac{\rho_{\mathcal{X}}(x_{1:n})}{\mu(x_{1:n})} \leq \frac{|\mathcal{X}| - 1}{2} \log_2 n + |\mathcal{X}| - 1. \tag{2}$$

As Equation 2 suggests however, performance of this particular coding technique can be poor for small values of n when $|\mathcal{A}|$ is much less than $|\mathcal{X}|$. This problem occurs often when

using context-based techniques for data compression. This is because, for many contexts, only a small subset of the full alphabet symbols are possible. For example, when modeling English text it is very rare to see any character other than the letter u immediately following the letter q. If we knew $\mathcal{A}$ in advance, we could code $x_{1:n}$ using $\rho_{\mathcal{A}}$, which from Equation 2 would of course give a redundancy no greater than

$$\frac{|\mathcal{A}|-1}{2}\log_2 n + |\mathcal{A}| - 1. \tag{3}$$

The Sequential Sub-alphabet estimator proposed by Tjalkens et al. [1993] provides a natural Bayesian solution to this dilemma. Rather than using the superset alphabet $\mathcal{X}$, their technique weights over the set of all possible Sequential Dirichlet Estimators whose alphabets are *subsets* of $\mathcal{X}$. This leads to an elegant algorithm that has a coding redundancy no more than

$$\log_2 |\mathcal{X}| + \log_2 \binom{|\mathcal{X}|}{|\mathcal{A}|} + \frac{|\mathcal{A}|-1}{2}\log_2 n + |\mathcal{A}| + 1, \tag{4}$$

when using a uniform prior over sub-alphabets. Unfortunately this method requires $O(|\mathcal{X}|)$ time to process each new symbol, and $O(|\mathcal{X}|)$ space. This can be prohibitive in situations where $|\mathcal{X}|$ is large. It would be better if the the time and space complexity were instead dependent on at most $|\mathcal{A}|$. This paper introduces a simple method, the Sparse Sequential Dirichlet Estimator, which achieves similar redundancy properties to the Sequential Sub-alphabet Estimator whilst being able to process each symbol in $O(1)$ time using at most $O(|\mathcal{A}|)$ space.

2 Preliminaries

We begin with some notation for data generating sources. An alphabet is a finite, non-empty set of symbols, which will denote as either $\mathcal{A}$ or $\mathcal{X}$. A binary string $x_1x_2\ldots x_n \in \mathcal{X}^n$ of length n is denoted by $x_{1:n}$. The prefix $x_{1:j}$ of $x_{1:n}$, $j \leq n$, is denoted by $x_{\leq j}$ or $x_{<j+1}$. The empty string is denoted by ϵ. The concatenation of two strings s and r is denoted by sr.

A probabilistic data generating source ρ is defined to be a sequence of probability mass functions $\rho_n : \mathcal{X}^n \to [0,1]$, for $n \in \mathbb{N}$, satisfying the constraint that

$$\rho_n(x_{1:n}) = \sum_{y\in\mathcal{X}} \rho_{n+1}(x_{1:n}y)$$

for all $x_{1:n} \in \mathcal{X}^n$, with base case $\rho_0(\epsilon) = 1$. As the meaning is always clear from the argument to ρ, we drop the subscripts on ρ from here onwards. Under this definition, the conditional probability of a symbol x_n given previous data $x_{<n}$ is defined as $\rho(x_n|x_{<n}) := \rho(x_{1:n})/\rho(x_{<n})$ if $\rho(x_{<n}) > 0$, with the familiar chain rule $\rho(x_{1:n}) = \prod_{i=1}^{n}\rho(x_i|x_{<i})$ now following.

A source code $c : \mathcal{X}^* \to \mathcal{X}^*$ assigns to each possible data sequence $x_{1:n}$ a binary codeword $c(x_{1:n})$ of length $\ell_c(x_{1:n})$. The typical goal when constructing a source code is to minimize the lengths of each codeword while ensuring that the original data sequence $x_{1:n}$ is always

recoverable from $c(x_{1:n})$. Given a data generating source μ, we know from Shannon's Source Coding Theorem that the optimal (in terms of expected code length) source code c uses codewords of length $-\log_2 \mu(x_{1:n})$ bits for all $x_{1:n}$. This motivates the notion of the *redundancy* of a source code c given a sequence $x_{1:n}$, which is defined as $r_c(x_{1:n}) := \ell_c(x_{1:n}) + \log_2 \mu(x_{1:n})$. Provided the data generating source is known, near optimal redundancy can essentially be achieved by using arithmetic encoding [Witten et al., 1987]. More precisely, using a_μ to denote the source code obtained by arithmetic coding using probabilistic model μ, the resultant code lengths are known to satisfy

$$\ell_{a_\mu}(x_{1:n}) < -\log_2 \mu(x_{1:n}) + 2, \tag{5}$$

for all $x_{1:n}$, which implies that the redundancy is always less than 2. In practice however, the true data generating source μ is typically unknown. The data can still be coded using arithmetic encoding with an alternate coding distribution ρ, however now we expect to use an extra $\mathbb{E}_\mu\left[\log_2 \mu(x_{1:n})/\rho(x_{1:n})\right]$ bits to code the random sequence $x_{1:n} \sim \mu$. From here onwards, we restrict our attention to that of specifying a good coding distribution.

3 Sparse Sequential Dirichlet Distribution

We now propose an adapted version of the Sequential Dirichlet Distribution, which will use less computational resources than the Sequential Sub-Alphabet Estimator, while still performing well in situations where $|\mathcal{A}|$ is much less than $|\mathcal{X}|$.

Definition 1. *Given an alphabet $\mathcal{X}$, for all $n \in \mathbb{N}$ and for all $x_{1:n} \in \mathcal{X}^n$, the Sparse Sequential Dirichlet distribution $\xi : \mathcal{X}^* \to (0, 1]$ is defined as*

$$\xi(x_{1:n}) := \prod_{i=1}^{n} \mathbb{I}[c(x_{1:i}) = 0]\, \alpha_i \frac{1}{|\mathcal{X}| - |U(x_{<i})|} + \mathbb{I}[c(x_{1:i}) > 0]\,(1 - \alpha_i) \frac{c(x_{1:i}) + \frac{1}{2}}{i + \frac{|U(x_{<i})|}{2} - 1} \tag{6}$$

where $c(x_{1:n}) := \sum_{i=1}^{n-1} \mathbb{I}[x_n = x_i]$, $U(x_{1:n}) := \{s \in \mathcal{X} : c(x_{1:n}s) > 0\}$ and $\alpha_i := \frac{1}{i}$ for $i \in \mathbb{N}$.

In the above, $U(x_{1:n})$ is simply the number of distinct symbols occurring in $x_{1:n}$. Furthermore, one can easily verify that ξ is a valid probability measure over finite but arbitrarily large strings whose symbols are from the alphabet $\mathcal{X}$.

Computational Properties. Given a sequence $x_{1:n} \in \mathcal{A}^n$, $\xi(x_{1:n})$ can be computed in $O(n)$ time, with $O(|\mathcal{A}|)$ space required to store the counts for the seen symbols. Furthermore, by using $\xi(x_n \,|\, x_{<n}) = \xi(x_{1:n})/\xi(x_{<n})$ in combination with the chain rule $\xi(x_{1:n}) = \xi(x_n \,|\, x_{<n})\xi(x_{<n})$, each symbol x_{n+1} can be processed in $O(1)$ time, leading to a straightforward incremental algorithm. As usual, numerical underflow issues can be addressed by storing all probability values in log-space.

Analysis. We now show that Sparse Sequential Dirichlet Coding using an alphabet of $\mathcal{X}$ performs well provided there exists an alphabet $\mathcal{A} \subset \mathcal{X}$ for which Sequential Dirichlet Coding performs well. Our goal will be to provide a redundancy bound which does not exhibit a linear dependence on $|\mathcal{X}|$.

Theorem 1. *Given alphabets $\mathcal{X}$ and $\mathcal{A}$ such that $\mathcal{A} \subseteq \mathcal{X}$, for all $n \in \mathbb{N}$, for all $x_{1:n} \in \mathcal{A}^n$, we have $-\log_2 \xi(x_{1:n}) \leq \log_2 n + |\mathcal{A}| \log_2 |\mathcal{X}| - \log_2 \rho_{\mathcal{A}}(x_{1:n})$.*

Proof. First note that since $|\mathcal{X}| \geq |\mathcal{X}| - |U(x_{<i})|$ and $|U(x_{1:n})| \leq |\mathcal{A}|$ for all $x_{1:n} \in \mathcal{A}^n$,

$$\xi(x_{1:n}) \geq \prod_{i=1}^{n} \mathbb{I}[c(x_{1:i}) = 0]\, \alpha_i \frac{1}{|\mathcal{X}|} + \mathbb{I}[c(x_{1:i}) > 0]\, (1 - \alpha_i) \frac{c(x_{1:i}) + \frac{1}{2}}{i + \frac{|\mathcal{A}|}{2} - 1}.$$

Now, noting that $\alpha_i = \frac{1}{i} \geq \frac{1}{2}/(i + |\mathcal{A}|/2 - 1)$ for all $i \in \mathbb{N}$, we get

$$\xi(x_{1:n}) \geq \prod_{i=1}^{n} \frac{c(x_{1:i}) + \frac{1}{2}}{i + \frac{|\mathcal{A}|}{2} - 1} \left(\mathbb{I}[c(x_{1:i}) = 0] \frac{1}{|\mathcal{X}|} + \mathbb{I}[c(x_{1:i}) > 0](1 - \alpha_i) \right).$$

Since there can be at most $|\mathcal{A}|$ new symbols, with the first symbol always being new, and

$$\prod_{1 \leq i \leq n \,:\, \mathbb{I}[c(x_{1:i}) > 0]} (1 - \alpha_i) \geq \prod_{i=2}^{n} (1 - \alpha_i),$$

we can conclude

$$\xi(x_{1:n}) \geq |\mathcal{X}|^{-|\mathcal{A}|} \prod_{i=1}^{n} \frac{c(x_{1:i}) + \frac{1}{2}}{i + \frac{|\mathcal{A}|}{2} - 1} \prod_{i=2}^{n} (1 - \alpha_i). \tag{7}$$

Now, simplifying the telescoping product and applying the definition of $\rho_{\mathcal{A}}$ (see Equation 1) to the right-hand side of Equation 7 gives $n^{-1}|\mathcal{X}|^{-|\mathcal{A}|}\rho_{\mathcal{A}}(x_{1:n})$. Hence,

$$-\log_2 \xi(x_{1:n}) \leq -\log_2 n^{-1}|\mathcal{X}|^{-|\mathcal{A}|}\rho_{\mathcal{A}}(x_{1:n}) = \log_2 n + |\mathcal{A}| \log_2 |\mathcal{X}| - \log_2 \rho_{\mathcal{A}}(x_{1:n}).$$

□

Thus, combining Theorem 1, Equation 5 and Equation 3, the overall coding redundancy of the Sparse Sequential Dirichlet Distribution is upper bounded by

$$\frac{|\mathcal{A}| + 1}{2} \log_2 n + |\mathcal{A}| \log_2 |\mathcal{X}| + |\mathcal{A}| + 1. \tag{8}$$

Discussion. A comparison of Equation 8 to Equation 2 suggests that the redundancy of Sparse Sequential Dirichlet Coding will be less than Sequential Dirichlet Coding when $|\mathcal{A}|$ is much smaller than $|\mathcal{X}|$. Furthermore, by applying the inequalities

$$|\mathcal{A}| \log_2 \tfrac{|\mathcal{X}|}{|\mathcal{A}|} \leq \log_2 \binom{|\mathcal{X}|}{|\mathcal{A}|} \leq |\mathcal{A}| \log_2 \tfrac{e|\mathcal{X}|}{|\mathcal{A}|}$$

to bound Equation 4, we can see that our redundancy bound for Sparse Sequential Dirichlet Coding is competitive with the redundancy bound for the Sequential Sub-alphabet estimator whenever $|X|$ is much larger than $|A|$, and worse when $|\mathcal{A}|$ is close to $|\mathcal{X}|$.

Method	Mean	Min	Max
ORACLE	185.048	12.3267	244.107
SDC($\mathcal{A}$)	193.953	21.368	243.718
SDC($\mathcal{X}$)	236.343	63.7581	286.108
SSD	210.844	23.7755	262.4
SSA	212.257	24.4074	262.928
SSD - SSA	-1.41272	-5.77022	0.366856

Table 1: Number of bits needed to encode 100 symbols when $|\mathcal{A}| = 5$ and $|\mathcal{X}| = 26$.

4 Numerical Experiments

We now present some numerical results for Sparse Sequential Dirichlet Coding, by comparing and contrasting our technique using the experimental framework described below.

Experimental Setup. Each different experiment consisted of evaluating the performance of 5 different coding distributions on synthetically generated data, for various choices of $\mathcal{A}$ and $\mathcal{X}$. The first technique, ORACLE, used the true underlying data generating distribution to code the data. This is of course the optimal coding distribution in expectation, and a natural baseline. The second and third techniques, SDC($\mathcal{A}$) and SDC($\mathcal{X}$), refer to using Sequential Dirichlet Coding using the alphabets $\mathcal{A}$ and $\mathcal{X}$ respectively. These two methods allow us to measure the impact of knowing and not knowing $\mathcal{A}$ in advance. SSD refers to our Sparse Sequential Dirichlet Coding technique. Finally, SSA refers to the Sequential Sub-Alphabet technique of Tjalkens et al. [1993].

To evaluate each particular combination of $\mathcal{A}$ and $\mathcal{X}$, 100,000 parameter vectors, $\mathbf{a}_i \in \mathbb{R}^{|\mathcal{A}|}$ for $1 \leq i \leq 100,000$, were sampled from a Symmetric Dirichlet Distribution using a concentration parameter of 1.0. These $\mathbf{a}_i$ were used to define a set of Categorical Distributions over the symbols in $\mathcal{A}$. Each Categorical Distribution was used once to generate a data sequence of 100 independent and identically distributed random symbols, which were then coded using each of the methods. The mean, min and max performance, measured in bits, for each different coding distribution on the generated data sequences was then summarised in Tables 1, 2 and 3. Additionally, the last line of each table measured how many extra bits Sparse Sequential Dirichlet Coding needed compared with Sequential Sub-Alphabet Coding.

Results. Table 1 and Table 2 compare the relative coding performance of Sparse Sequential Dirichlet Coding when $|\mathcal{A}|$ is much less than $|\mathcal{X}|$. In both situations we see that the Sparse Sequential Dirichlet technique is on average slightly superior to the Sequential Sub-Alphabet method, and never worse by more than 1.32 bits. Both techniques performed significantly better than the Sequential Dirichlet Coding method which used the alphabet $\mathcal{X}$. This is consistent with the redundancy bounds we presented earlier. Lastly, Table 3 gives an example of what can happen when the sparsity assumption doesn't apply. Here we see that the Sparse Sequential Dirichlet method is outperformed by all other techniques, though not

Method	Mean	Min	Max
ORACLE	278.363	131.529	340.359
SDC($\mathcal{A}$)	293.969	146.716	349.882
SDC($\mathcal{X}$)	492.284	345.031	548.197
SSD	349.169	181.365	412.766
SSA	350.473	187.604	411.656
SSD - SSA	-1.30374	-7.4791	1.3234

Table 2: Number of bits needed to encode 100 symbols when $|\mathcal{A}| = 10$ and $|\mathcal{X}| = 256$.

Method	Mean	Min	Max
ORACLE	360.053	234.325	422.467
SDC($\mathcal{A}$)	382.911	258.392	440.005
SDC($\mathcal{X}$)	396.527	272.007	453.62
SSD	410.573	277.942	476.754
SSA	397.344	271.68	456.234
SSD - SSA	13.2292	0.446927	20.5248

Table 3: Number of bits needed to encode 100 symbols when $|\mathcal{A}| = 18$ and $|\mathcal{X}| = 26$.

by a large margin.

Discussion. In light of its superior computational properties, our results suggest that the Sparse Sequential Dirichlet technique is a good alternative to the Sequential Sub-Alphabet method whenever $|\mathcal{A}|$ is much less than $|\mathcal{X}|$. If issues of computation or limited memory are not an issue, the Sequential Sub-Alphabet method is to be preferred due to its better performance when $|\mathcal{A}|$ is not much less than $|\mathcal{X}|$.

5 Conclusion

This short paper has described a simple and efficient coding technique for multi-alphabet memoryless sources. It provably works well when only a small subset of possible alphabet symbols are expected to occur in any given data sequence. As future work, it would be interesting to explore the applicability of this technique as a building block within more sophisticated context modeling techniques.

Acknowledgements The authors would like to thank Kee Siong Ng and Marc Bellemare for comments that helped improve this paper.

References

R. Krichevsky and V. Trofimov. The performance of universal encoding. *Information Theory, IEEE Transactions on*, 27(2):199–207, 1981.

Tjalling J. Tjalkens, Yuri M. Shtarkov, and Frans M. J. Willems. Sequential weighting algorithms for multialphabet sources. In *6th Joint Swedish-Russian Int. Worksh. Inform. Theory*, pages 22–27, 1993.

Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30:520–540, June 1987. ISSN 0001-0782.