Milestone Report for EE274 Bits-back coding and compression of data with permutation invariance

Yifei Wang

1 Introduction

In this project, we will investigate the compression of multisets where the relative order of the symbols are irrelevant, for instance, collections of files. Severo et al. (2022) proposes a new compression algorithm which saves computational cost by avoiding encoding the order between symbols. We follow this paper as the context and background.

2 Asymmetric Numeral Systems and Bits-Back Coding

We first briefly review asymmetric numeral systems (ANS), which serves as the basis for bits-back coding. For a symbol $x \in \mathcal{A}$ with mass function D, the encode and decode functions form an inverse pair:

$$\begin{aligned}\texttt{encode} :& \mathbb{N} \times \mathcal{A} \to \mathbb{N}, (s, x) \to s',\\ \texttt{decode} :& \mathbb{N} \to \mathbb{N} \times \mathcal{A}, s' \to (s, x).\end{aligned} \tag{1}$$

Here $\log s' \approx \log s + \log 1/D(x)$. To implement `encode` and `decode`, the quantized distribution D must be specified by a precision parameter N together with two lookup functions:

$$\begin{aligned}\texttt{forward\_lookup} &: x \to (c_x, p_x),\\ \texttt{reverse\_lookup} &: i \to (x, c_x, p_x).\end{aligned} \tag{2}$$

Here c_x and p_x are quantized cumulative distribution function and probability mass satisfying

$$\frac{p_x}{N} = D(x), \quad c_x = \sum_{y<x} p_x. \tag{3}$$

The `reverse_lookup` takes the input $i \in \{0, \dots, N\}$ and it must output (x, c_x, p_x) such that $i \in [c_x, c_x + p_x)$.

The key to apply ANS to bits-back coding is the observation that we can use `decode` for a quantized distribution D as an invertible sampler. To be specific, the ANS state s is thus used as a random seed. By executing `decode` to s, it removes $-\log D(x)$ bits from the state s. In short, the random seed is slowly consumed as symbols are sampled.

ANS can be directly applied to compress a multiset. Consider a multiset $\mathcal{M} = \{x_1, \dots, x_m\}$, where $m = |\mathcal{M}|$. We can view $x_1, \dots, x_m$ as a sequence of i.i.d. symbols $x^m = x_1 \dots x_m$ from a distribution D over alphabet $\mathcal{A}$. We can compress x^m by initializing the state $s_0 = 0$ and recursively update $s_n = \texttt{encode}(s_{n-1}, x_n)$ with D. The finial ANS state s_m will be approximately

$$\log s_m \approx \sum_{n=1}^{m} \log 1/D(x_n).$$

Now, we describe the algorithm of bits-back coding as follows. To encode, we store a multiset of remaining symbols $\mathcal{M}_n$. Firstly, we set $\mathcal{M}_1 = \mathcal{M}$. In the n-th iteration, we sample z^n without replacement from $\mathcal{M}_n$ and encode it. Let z^{n-1} be the sequence of previously sampled symbols. Then, we note that

$$P(z_n|z^{n-1}) = \frac{\mathcal{M}_n(z_n)}{|\mathcal{M}_n|}. \tag{4}$$

To sample z_n decreases the state s_n, while the encoding step increases s_n. To be specific, the number of bits of the state increase at the n-th round is approximately

$$\Delta_n = \log \frac{D(z_n)}{P(z_n|z^{n-1})}. \tag{5}$$

The total increase in the state is

$$\sum_{n=1}^{m} \Delta_n = \sum_{n=1}^{m} \log \frac{D(z_n)}{P(z_n|z^{n-1})} = \log \prod_{z\in\mathcal{M}} D(z)^{\mathcal{M}(z)} - \log \frac{m!}{\prod_{z\in\mathcal{M}} \mathcal{M}(z)!} = \log P(\mathcal{M}). \tag{6}$$

This is exactly the negative information content of the multiset. Suppose that we start with $\log s_0 \approx \sum_{n=1}^{m} \log 1/D(x_n)$. Then, we have

$$\log s_m \approx -\log \frac{m!}{\prod_{z\in\mathcal{M}} \mathcal{M}(z)!}. \tag{7}$$

In summary, by using the bits-back coding to compress multisets, we can save $-\log P(\mathcal{M})$ bits compared to directly apply ANS to compress multisets.

To efficiently implement the invertible sampling without replacement, we utilizes the binary search tree (BST). In summary, we describe the multiset encoder and decoder in Algorithm 1.

Algorithm 1 Multiset Encoder and Decoder for bits-back coding.

```
1:  function MultisetEncode($\mathcal{M}$)
2:      Initiate a random state $s_0$. Set $m = |\mathcal{M}|$ and $\mathcal{M}_1 = \mathcal{M}$.
3:      for $n = 1, \dots, m$ do
4:          $(s'_n, z_n) = \texttt{decode}(s_{n-1})$ with $P(\cdot|\mathcal{M}_n)$        ▷ $O(\log m)$
5:          $\mathcal{M}_{n+1} = \mathcal{M}_n / \{z_n\}$        ▷ $O(\log m)$
6:          $s_n = \texttt{encode}(s'_n, z_n)$ with $D(\cdot)$        ▷ $O(D_{\text{Encode}})$
7:      end for
8:      return $s_m$
9:  end function
10: function MultisetDecode($s_m$)
11:     Initialize $\mathcal{M}_{m+1} = \varnothing$.
12:     for n=m,...,1 do
13:         $(s'_n, z_n) = \texttt{decode}(s_n)$ with $D(\cdot)$        ▷ $O(D_{\text{Decode}})$
14:         $\mathcal{M}_n = \mathcal{M}_{n+1} \cup \{z_n\}$        ▷ $O(\log m)$
15:         $s_{n-1} = \texttt{encode}(s'_n, z_n)$ with $P(\cdot|\mathcal{M}_n)$.        ▷ $O(\log m)$
16:     end for
17:     return $\mathcal{M}_1$
18: end function
```

References

Severo, D., Townsend, J., Khisti, A., Makhzani, A., and Ullrich, K. (2022). Compressing multisets with large alphabets. In *2022 Data Compression Conference (DCC)*, pages 322–331. IEEE.