Smooth Representation of Rankings

(Invited Paper)

Arya Mazumdar
Department of ECE
University of Minnesota– Twin Cities
Minneapolis, MN 55455
email:arya@umn.edu

Olgica Milenkovic
Department of ECE
University of Illinois
Urbana, IL 61801
email: milenkovic@illinois.edu

Abstract—An encoding of data for digital storage is called smooth, if, for any small change in the raw data, only a proportionately small change has to be made in the encoded data. In this paper, we consider the problem of smooth encoding for ordinal data, i.e., rankings. It is shown that, simple efficient smooth encoding for storage (as well as compression) of rankings is possible with respect to the well-known Kendall τ metric on permutations.

I. INTRODUCTION

Ordinal data conveys information about the relational properties of data instead of their absolute numerical values. In various applications, such as in social sciences or economics, the true value of data is frequently missing, erroneous or misleading, while on the other hand relative ordering or ranking of data is easily computable.

Ranking of data is represented usually by a permutation. However digital storage in databases with conventional media is done by writing integers from a discrete alphabet (most frequently, binary) in the memory. Hence, for storage purposes, ordinal datasets, namely rankings, are mapped to binary vectors. This encoding is straightforward in most of the cases, and it is well-known that to store a permutation of $n$ elements, one needs about $n \log_2 n$ bits of storage.

In many cases, the complete ranking is too big to store, and only some partial information might be of interest. For example, in a recommendation system for restaurants (such as YELP.com), the ranking at the top matters more than the intricacies at the bottom of the list. For these reasons, a ranking may be compressed allowing some distortion in recovery. In such cases, the amount of storage needed, and the error in recovery allowed, naturally present a rate-distortion theory of permutations [1].

Most of the ordinal data, generated from recommender systems and search engines and/or stored in distributed storage systems (such as, cloud storage), undergoes small, but frequent changes which have to be accounted for in the representation/compressed format. Smooth representation techniques satisfy the need for fast processing of such volatile data. Updating stored digital data requires access and time, and therefore bandwidth and energy. If the changes in ordinal data are large, then it may be inevitable to update a large number of stored symbols. Nevertheless, it is desirable to have sublinear (in storage amount) updates in encoded data for comparably small changes in the original ordinal data, especially in BigData applications. This problem, under the name of update-efficient error-correcting codes, was recently addressed in [2] in the context of binary data and channel coding. For compression of standard binary data, the smoothness of lossless compression was studied in [3].

It is not clear in advance if smooth compression is plausible for ordinal data, given that near-optimal sorting and compression algorithms map points in $S_n$ to points $\{0,1\}^n$ that obey a near-uniform distribution and may lie at a large average Hamming distance from each other. In this paper, we show that such smooth representation and compression is possible for permutations with respect to the very popular and useful Kendall τ distance and its generalizations. In the following section, we present some preliminaries regarding the space of permutations. In Sections III and IV, we, respectively for the lossless and the lossy cases, show the smoothness property of a storage algorithm.

II. PRELIMINARIES

A permutation $\sigma : [n] \to [n]$, that is, for any $i,j \in [n], i \neq j$, one has $\sigma(i) \neq \sigma(j)$. We let $S_n$ denote the set of all permutations of the set $[n]$, i.e., the symmetric group of order $n!$. For any $\sigma \in S_n$, we write $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$, the vector representation of a permutation, where $\sigma(i)$ is the image of $i \in [n]$ under the permutation $\sigma$. The inverse $\sigma^{-1}$ of a permutation $\sigma$ is a permutation in which an element and the position that it occupies are exchanged.

To store a permutation on $n$ elements, one needs $\log_2(n!) \approx n \log_2(n/e)$ bits. However, to directly store the vector representation naively, one ends up using $n \log_2 n$ bits. To get rid of this extra $O(n)$ bits, we may encode the permutation to its Lehmer code first.

A permutation $\sigma \in S_n$ may be uniquely encoded to its Lehmer code (also called the inversion vector), $x_\sigma \in \mathcal{H}_n = [0,1] \times [0,2] \times \cdots \times [0,n-1]$, where $x_\sigma(i) = |\{j \in [n] : j < i+1, \sigma^{-1}(j) > \sigma^{-1}(i+1)\}|$, $i = 1, \ldots, n-1$. In words, $x_\sigma(i), i = 1, \ldots, n-1$ is the number of inversions (pairs out of order) in the permutation $\sigma$ for which $i+1$ is the first element. For instance, we have...
It is well known that the Lehmer code is bijective, and simple reconstruction algorithms are known [4]. They are of particular interest for compression, given that they perform a zero-distortion transform from the domain of ordinals to the domain of quantitative data.

Now, the ith coordinate of a Lehmer code can be stored using \(\log_2(i + 1)\) bits without any loss. Hence, a total of \(\sum_{i=1}^{n-1} \log_2(i + 1) \approx \log_2(n!)\) bits are used for storage.

An representation \(f: S_n \rightarrow \{0, 1\}^m\) is termed as a source code here. When \(m \geq \log_2(n!}\) the representation is lossless, and otherwise the representation is a lossy compression.

**Definition 1:** A source code \(f: S_n \rightarrow \{0, 1\}^m\) is said to be \((u, t)\)-smooth with respect to a distortion measure \(d\) if, for any \(\pi, \sigma \in S_n\), \(d(\pi, \sigma) \leq u\) implies \(d_f(f(\pi), f(\sigma)) \leq t\), where \(d_f(\cdot, \cdot)\) denotes the Hamming distance.

The metric on permutation that is of interest to us in this paper is the Kendall \(\tau\) metric. Kendall \(\tau\) distance is a natural metric on permutations, that was introduced in [5] for application in statistics, and then adapted as a suitable measure for various application in computer science and bioinformatics (e.g., [6], [7]) and most recently in error-correcting codes [8], [9].

**Definition 2:** The Kendall \(\tau\) distance \(d_\tau(\cdot, \cdot)\) between any two permutations is the minimum number of pairwise adjacent swaps needed to convert one to other. Formally, let \(I(\sigma)\) denote the number of inversions in \(\sigma \in S_n\). For any two permutations \(\pi, \sigma\),

\[
d_\tau(\pi, \sigma) = I(\sigma \pi^{-1})
\]

\[
= \|\{(i, j) : \pi^{-1}(i) > \pi^{-1}(j), \sigma^{-1}(i) < \sigma^{-1}(j)\}\|
\]

In what follows, we consider a \((u, t)\)-smooth codes in the context of Kendall metric. A practical generalization of \(d_\tau\) called weighted Kendall metric has recently been proposed in [10], [11]. The generalization of the result presented in this paper to weighted Kendall metric will appear in the full version of this paper.

Finally, other interesting and practical distances on permutations, such as the Ulam metric [12], may be considered to construct smooth codes 1.

**III. LOSSLESS REPRESENTATION**

In this section, we exhibit a smooth mapping for the Kendall \(\tau\) distance which illustrates the approach to be pursued for more general distance measures. The ideas behind the techniques were used in the context of constructing error-correcting codes in [14].

Define the \(\ell_1\) distance function on \(\mathcal{H}_n\) as

\[
d_1(x, y) = \sum_{i=1}^{n-1} |x(i) - y(i)|, \quad (x, y \in \mathcal{H}_n)
\]

with the computations performed over the set of integers. For instance, if \(\sigma_1 = (2, 1, 4, 3)\) and \(\sigma_2 = (2, 3, 4, 1)\), then the Lehmer codes of \(\sigma_1\) and \(\sigma_2\) equal \(x_{\sigma_1} = (1, 0, 1)\) and \(x_{\sigma_2} = (1, 1, 1)\). To compute the distance \(d_1(\sigma_1, \sigma_2)\), we note that \(\sigma_1^{-1} = \sigma_1\) and so \(I(\sigma_2 \sigma_1^{-1}) = I((1, 4, 3, 2)) = 3\). Observe that the mapping \(\sigma \mapsto x_\sigma\) is a weight-preserving bijection between \(S_n\) and \(\mathcal{H}_n\), although it is not distance preserving. Indeed, \(d_\tau(\sigma_1, \sigma_2) = 3\) while \(d_1(x_{\sigma_1}, x_{\sigma_2}) = 1\). It can actually be shown [8] that \(d_1(\sigma_1, \sigma_2) \geq d_\tau(x_{\sigma_1}, x_{\sigma_2})\).

We make use of the binary Gray code, which is a mapping \(\phi_s\) from the ordered set of integers \([0, 2^s - 1]\) to \([0, 1]^s\), with the property that the images of two successive integers differ in exactly one bit. Suppose that \(b_{s-1}b_{s-2} \ldots b_0\), \(b_i \in \{0, 1\}, 0 \leq i < s\), is the binary representation of an integer \(u \in [0, 2^s - 1]\). By definition, set \(b_s = 0\) and construct \(\phi_s(u) = (g_{s-1}, g_{s-2}, \ldots, g_0)\) where

\[
g_j = (b_1 + b_{j+1}) \mod 2, \quad j = 0, 1, \ldots, s - 1.
\]

The Gray map for the first 10 integers looks as follows:

\[
\begin{align*}
0 & \rightarrow 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & \rightarrow 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
2 & \rightarrow 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
3 & \rightarrow 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
4 & \rightarrow 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
5 & \rightarrow 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
6 & \rightarrow 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
7 & \rightarrow 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
8 & \rightarrow 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
9 & \rightarrow 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\end{align*}
\]

Note the “reflective” nature of the map: the last 2 bits of the second block of four are a reflection of the last 2 digits of the first block with respect to the horizontal line; the last 3 bits of the second block of eight follow a similar rule, and so on.

It is straightforward to see that \(|1 - j| \geq d_1(\phi_s(1), \phi_s(j))\) for any two \(i, j \in \{0, 1, \ldots, 2^s - 1\}\).

Let us now describe the representation mapping for permutations as follow. Assume that one is given \(\sigma \in S_n\), with Lehmer code \(x_\sigma = (x_\sigma(1), \ldots, x_\sigma(n-1))\). Let the representation map \(f: S_n \rightarrow \{0, 1\}^m\) be of the form

\[
f(\sigma) = (\phi_{m_1}(x_\sigma(1)), \ldots, \phi_{m_{n-1}}(x_\sigma(n-1))),
\]

where \(m_i = \lfloor \log_2 i \rfloor\) and \(m = \sum_{i=1}^{n-1} m_i < \log_2(n!) + n\). The underlying mapping is \((u, t)\)-smooth for any integer \(u\). In addition, the overhead of the method compared to the optimal (not necessarily smooth) compression scheme is at most \(n\) bits.

The above analysis works when \(u = o(n \log n)\). But, the largest possible value of the Kendall distance is \(\binom{n}{2}\). Indeed, this maximum occurs when two permutations are exactly in the reverse order in the vector notation (e.g., \((1, 2, \ldots, n)\) and \((n, n-1, \ldots, 1)\)). Hence a small change in ranking may mean

\footnote{Another popular metric, called the Spearman’s footrule, does not lead to any interesting question beyond the case for Kendall \(\tau\) metric, because of their equivalence within a constant factor [13].}
clearly, $\sum_{i=1}^{n-1} w_i \leq w$, and from Lemma 1,
\[ d_H(\phi_{m_1}(x_\sigma(i)), \phi_{m_1}(x_\pi(i))) \leq [\log_2 w_i] + 1, \]
whenever $w_i \geq 1$. Hence, assuming total number of nonzero $w_i$s to be $N$,
\[ d_H(f(\sigma), f(\pi)) \leq \sum_{i=1}^{n-1} (\log_2 w_i + 1) \leq \sum_{i=1}^{n-1} \log_2 w_i + 2N \leq N \log_2 \frac{\sum_{i=1}^{n-1} w_i}{N} + 2N \leq (n-1) \left( \frac{\log_2 w}{n-1} + 2 \right), \]
where we have used the concavity of the log function and Jensen’s inequality.

From the above, we claim that the representation of permutations by $f$ defined above is smooth everywhere. First of all, the mapping is $(u,t)$-smooth, with $t = \min \left\{ u, |n-1| \left( \frac{\log_2 \frac{w}{n-1} + 2 \right) \right\}$. In particular, when $u = O(n)$, $t = u$. But, recall, Kendall distance can be as large as $\Omega(n^2)$. The above theorem tells us when $u = \Omega(n^{1+\delta})$, $t = \delta n \log_2 n + O(\delta n) = \delta m + o(m)$, for any $\delta > 0$.

In the next section we show that, not only the lossless representation, but also the scalar quantization lossy source coding algorithm [1, Sec. V] for compression of rankings, is also smooth.

IV. LOSSY COMPRESSION

Assume, we want to construct a $(u,t)$-smooth code that compresses the permutations with a worst-case distortion guarantee $D$. In the lossy compression, a source code $\mathcal{C} \in S_n$ is a set of permutation such that, given any $\sigma \in S_n$, there exists an $\pi \in \mathcal{C}$ such that $d_\pi(\sigma, \pi) \leq D$. The elements of $\mathcal{C}$ are then represented and stored in binary. A lossy compression algorithm, a simple scalar quantization, is presented in [1, Sec. V].

The algorithm is a generalization of the lossless representation above. Given any permutation $\sigma$, in the first step of the algorithm, its Lehmer code $x_\sigma$ is found. Then each coordinate of $x_\sigma$ is independently quantized with uniform quantization levels. For example, the $i$th coordinate may take value in $\{0,1,\ldots,i\}$. We divide this coordinate in $\ell_i$ different levels with any two levels uniformly separated by $2D_i$. We must have, $D_i = \frac{2(i+1)}{(n+1)(n-2 i)}$, so that, $\sum_{i=1}^{n-1} D_i = D$. For details, we refer the reader to [1] ².

Suppose this compression algorithm is $(u,t)$-smooth. We will find the values of $t$ given $u$ next. Suppose, $u_i$ is the absolute difference in each coordinate of the Lehmer code from a permutation $\sigma$ and its updated version $\sigma'$ (that is, $|x_\sigma(i) - x_{\sigma'}(i)| = u_i$). Clearly, $\sum_{i=1}^{n-1} u_i = d_1(x_\sigma, x_{\sigma'}) \leq 2$.

The distortion here is measured as the $\ell_1$ distance of Lehmer code, and not the Kendall $\tau$ distance. However, it was shown in [15] recently that this distance is very close to Kendall $\tau$ distance and the average case rate-distortion properties are same for these two distances.
\(d_{\tau}(\sigma, \sigma') = u\). Now if \(q\) is the quantizer function in the \(i\)th coordinate, then clearly, \(|q(x_\sigma(i)) - q(x_{\sigma'}(i))| \leq u_1 + 2D_1\), or difference in each coordinate is upper bounded by \(u_1 + 2D_1\).

But from Lemma 1,

\[
\begin{align*}
t &\leq \sum_{i=1 \text{ to } n-1: u_i + 2D_i \neq 0} \left[ \log_2 (u_i + 2D_i) \right] + 1 \\
&\leq \sum_{i=1 \text{ to } n-1: u_i + 2D_i \neq 0} \log_2 (u_i + 2D_i) + 2(n - 1) \\
&\leq (n - 1) \log_2 \left( \frac{\sum_{i=1}^{n-1} (u_i + 2D_i)}{n - 1} \right) + 2(n - 1) \\
&\leq (n - 1) \left( \log_2 \frac{u + 2D}{n - 1} + 2 \right).
\end{align*}
\]

And, hence, conclusions similar to the previous section can be drawn. In particular, when \(u = O(n^{1+\delta})\) for some \(0 < \delta \leq 1\), and \(u > D\), then the amount of update is only \(\delta n \log_2 n + O(n)\).

**Acknowledgement:** A. Mazumdar’s work was supported in part by NSF CCF1318093 and a grant from University of Minnesota.

**REFERENCES**


