Learned Monotone Bucketization as Partial Flattening

Jonathan R. Landers

Abstract

This short note sketches a learning-augmented extension of *Algorithms as Geodesics: Flattening Entropy and Partial Curvature Removal in Sorting*. In the base framework, an ordered bucket map removes a measurable block of sorting entropy before exact in-bucket comparison cleanup begins. The extension proposed here is to learn the monotone preprocessing map itself from calibration data, so that bucket occupancies are as close to balanced as possible under the ambient key distribution. The resulting runtime admits a simple decomposition: ideal partial flattening plus a KL-type regret term, together with the cost of learning.

1. Context

The main paper shows that ordered bucketization gives an exact entropy decomposition. If

$$\beta : \mathcal{K} \to \{1, \dots, B\}$$

is an ordered bucket map and the induced bucket occupancies are $n_1, \dots, n_B$, then the residual sorting entropy is

$$F_{\mathrm{res}}(x; \beta) = \sum_{j=1}^{B} \log(n_j!),$$

while the flattening gain is

$$G(x; \beta) = \log(n!) - \sum_{j=1}^{B} \log(n_j!) = \log \frac{n!}{\prod_{j=1}^{B} n_j!}.$$

Because the logarithms are natural, these quantities are measured in *nats*, so the gain is a literal amount of information removed by preprocessing. Balanced occupancies maximize this gain. The natural next step is therefore not to replace sorting by learning, but to learn a good monotone preprocessing chart that removes as much sorting entropy as possible before exact local refinement.

2. Learned Monotone Bucketization

Let $\mathcal{K}$ be a totally ordered key space and let P be an unknown distribution on $\mathcal{K}$. From calibration data $z_1, \dots, z_m \sim P$, learn a monotone score

$$s : \mathcal{K} \to [0, 1],$$

intended to approximate the CDF $F(x) = P(X \le x)$. For a chosen bucket count B, define

$$\widehat{\beta}_B(x) = 1 + \min\big(B - 1, \lfloor B\, s(x) \rfloor\big).$$

Definition 1. The map $\widehat{\beta}_B$ is a *statistically learned monotone bucketization.* Since s is monotone, $\widehat{\beta}_B$ is an ordered bucket map:

$$\widehat{\beta}_B(x) < \widehat{\beta}_B(y) \implies x < y.$$

This yields the hybrid procedure:

1. learn s from calibration data;
2. bucket the input by $\widehat{\beta}_B$;
3. sort exactly within each bucket;
4. output the buckets in bucket order.

Proposition 2. *The learned partial-flattening sort is exact.*

Proof. Because $\widehat{\beta}_B$ is ordered, every element in bucket j is less than or equal to every element in bucket k whenever $j < k$. Hence all unresolved order is internal to buckets, and exact in-bucket sorting yields the exact global sorted order. □

3. Complexity and Bucketization Regret

For an input $x = (x_1, \ldots, x_n)$, let

$$n_j = \#\{i : \widehat{\beta}_B(x_i) = j\}, \qquad p_j = \frac{n_j}{n}, \qquad u = \Big(\frac{1}{B}, \ldots, \frac{1}{B}\Big).$$

The ordered-bucketing theorem from the main paper gives residual entropy

$$F_{\mathrm{res}}(x; \widehat{\beta}_B) = \sum_{j=1}^{B} \log(n_j!).$$

If the in-bucket refinement uses mergesort, the total runtime satisfies the following bound.

Theorem 3. *Let $C_{\mathrm{learn}}(m, B)$ denote the cost of producing the score s from calibration data. Then*

$$T_{\widehat{\beta}_B}(x) \le C_{\mathrm{learn}}(m, B) + O(n + B) + \sum_{j=1}^{B} O(n_j \log n_j).$$

Equivalently,

$$T_{\widehat{\beta}_B}(x) = C_{\mathrm{learn}}(m, B) + O\Big(n \log \frac{n}{B} + nD_{\mathrm{KL}}(p\|u) + n + B \log n\Big).$$

Proof. The first bound is linear-time bucketing plus exact comparison sorting inside each bucket. For the second, write $n_j = np_j$. Then

$$\sum_{j=1}^{B} n_j \log n_j = n \log n + n \sum_{j=1}^{B} p_j \log p_j = n \log \frac{n}{B} + nD_{\mathrm{KL}}(p\|u).$$

Replacing exact factorial terms by Stirling's approximation contributes only lower-order $O(n+B \log n)$ terms. □

This motivates the definition

$$R_B(x;\widehat{\beta}_B) := nD_{\mathrm{KL}}(p\|u),$$

which may be read as the *bucketization regret*. The term $n\log(n/B)$ is the cost of ideal balanced partial flattening, while R_B is the excess cost caused by imperfect learned bucketization. In this language, the learning problem is to construct the cheapest monotone chart whose occupancy profile is as close to uniform as possible.

Equivalently, one can view the preprocessing stage as solving

$$\max_{\beta\in\mathcal{M}_{\mathrm{mono}}}\Big[\mathbb{E}\,G(X_{1:n};\beta)-\lambda C_{\mathrm{pre}}(\beta)\Big],$$

over a class of monotone bucket maps. The conceptual slogan is:

sorting after learned bucketization = ideal partial flattening + KL regret + learning cost.