Exploitable Internal Redundancy

From Local Coincidences to the Cost of Accessing Structure

Jonathan R. Landers

August 2026

Abstract

A computation can be large because its input is large, or because it performs work that the structure of the problem does not require. We trace increasingly strong reductions. For $S = \sum_{i=1}^{N}(ax_i + by_i)$, with fixed weights a, b, local coincidences remove arithmetic only when they are cheap enough to detect. Global linearity is stronger: it factors the computation through two sums, asymptotically halves primitive arithmetic, and amortizes the scan across repeated queries. For a function family $\mathcal{F} = \{f_\theta\}$ inducing additive queries $F_\theta(D) = \sum_i f_\theta(z_i)$, the exact summary rank $r(\mathcal{F}) = \dim \operatorname{span} \mathcal{F}$ is precisely the minimum number of scalar additive summaries required for linear exact recovery. Input-side structure can then change the order: a d-feature predictor on an accessible k-dimensional subspace costs $\Theta(k)$, not $\Theta(d)$. But dimension is not runtime. We define an intrinsic prediction rank s and a raw-coordinate accessibility τ, prove $s \le \tau \le k$, and show that every such gap occurs. Thus even a one-dimensional answer may require reading k stored coordinates. The unifying principle is to factor through a smaller space while counting the cost of reaching it.

1 The question: when is work inside a computation redundant?

Suppose one is handed the sum

$$S = \sum_{i=1}^{N}(ax_i + by_i), \tag{1}$$

where $N \in \mathbb{N}$ is the number of indexed pairs, $a, b \in \mathbb{R}$ are fixed weights (in the motivating example, percentages), and $x_i, y_i \in \mathbb{R}$. The naive implementation performs the same small arithmetic pattern at every index: multiply twice, add the two weighted terms, then accumulate.

The first observation is almost embarrassingly simple. Whenever $x_i = y_i$,

$$ax_i + by_i = (a + b)x_i. \tag{2}$$

One multiplication has disappeared. Nothing asymptotic has changed, yet a real unit of work has vanished because the input contained a local relation. This is the seed of the present note:

> *Input size measures how much data is present. It need not measure how much distinct computation the data requires.*

This perspective lies near several established traditions, from compiler optimization and precomputed aggregation to statistical sufficiency and compressed representations. Our aim is narrower and arithmetic:

begin with the smallest example and let each structural view expose the limitation of the previous one—from local savings, to global summaries, to lower-dimensional prediction, and finally to the gap between intrinsic dimension and the cost of accessing it.

Takeaway. The apparent size of an input can overstate the distinct work its structure actually requires.

2 Local redundancy and an instance parameter

Let

$$\gamma = \#\{i \in \{1, \dots, N\} : x_i = y_i\} \tag{3}$$

be the number of locally factorable indices. Assume first that these indices are already known, so that detection is free, and precompute $a + b$ when $\gamma > 0$.

At a nonredundant index we use two multiplications and one within-term addition. At a redundant index we use one multiplication. Accumulating N terms requires $N - 1$ further additions. Thus, for $\gamma > 0$, the arithmetic count is

$$T_{\text{local}}(N, \gamma) = (2N - \gamma) + (N - \gamma) + (N - 1) + 1 = 4N - 2\gamma, \tag{4}$$

whereas the direct computation uses

$$T_{\text{direct}}(N) = 2N + (2N - 1) = 4N - 1. \tag{5}$$

The dependence on γ is linear: every additional local coincidence removes two primitive operations under this simple unit-cost model, apart from the one-time setup.

This is an instance-sensitive statement, not an asymptotic improvement. Both procedures remain $O(N)$. The point is that two inputs of the same length can require different arithmetic because their internal relations differ.

2.1 Detection is part of the algorithm

The previous count is deliberately optimistic. If equality must be tested at every index, redundancy has a discovery cost. Let c_{det} be the cost of testing one location and c_{save} the work saved by exploiting one redundant location. A generic cost model is

$$T_{\text{exploit}} = T_{\text{base}} + c_{\text{det}} N - c_{\text{save}} \gamma, \tag{6}$$

where T_{base} is the cost of ignoring the local coincidences and T_{exploit} is the cost of detecting and using them. Writing $\rho = \gamma / N$ for the redundancy density, exploitation helps exactly when

$$\rho > \frac{c_{\text{det}}}{c_{\text{save}}}. \tag{7}$$

This elementary threshold is conceptually important. Redundancy is not a computational resource merely because it exists; it must be *cheap enough to expose*. Compiler optimization makes the same practical distinction between available reuse and profitable reuse [1].

At this point one might try to improve equality detection, cluster repeated values, or search for more elaborate local relations. But Eq. (1) contains a stronger structure that makes all of that unnecessary.

Takeaway. Local redundancy saves work only when finding it costs less than using it saves.

3 The turn: the important redundancy is global

Because a and b do not depend on i, linearity gives

$$S = a\sum_{i=1}^{N} x_i + b\sum_{i=1}^{N} y_i. \tag{8}$$

Define

$$X = \sum_i x_i, \qquad Y = \sum_i y_i. \tag{9}$$

Then the entire input is reduced, for the purpose of this computation, to the pair (X, Y), and

$$S = aX + bY. \tag{10}$$

The striking point is not the algebra; it is what happened to γ. The global summary works even when $\gamma = 0$. We began by searching for repeated *values*. The better question was to search for repeated *computational roles*.

The operation counts make this visible. The direct method uses $2N$ multiplications and $2N-1$ additions. The summary method uses only two multiplications, together with $2N-1$ additions: $2N-2$ to form X, Y, and one to combine $aX + bY$. Hence

$$T_{\text{summary}}(N) = 2N + 1 \tag{11}$$

under equal primitive costs, and

$$\frac{T_{\text{direct}}(N)}{T_{\text{summary}}(N)} = \frac{4N-1}{2N+1} \longrightarrow 2. \tag{12}$$

The asymptotic class is still linear, but asymptotically half the primitive arithmetic is removed.

More generally, if multiplication and addition cost $c_\times$ and c_+, respectively, then

$$T_{\text{direct}} = 2Nc_\times + (2N-1)c_+, \tag{13}$$

$$T_{\text{summary}} = 2c_\times + (2N-1)c_+, \tag{14}$$

so, defining $\Delta T = T_{\text{direct}} - T_{\text{summary}}$, the exact saving is

$$\boxed{\Delta T = 2(N-1)c_\times.} \tag{15}$$

The improvement is therefore largest precisely when the operation moved outside the data loop is expensive.

Takeaway. Repeated computational roles can yield larger savings than repeated input values.

4 From two sums to a general exact summary

The preceding example suggests a general construction. Let $D = (z_1, \ldots, z_N)$ be a dataset with observations in a domain $\mathcal{Z}$, let Θ be a parameter set, and consider real-valued per-observation functions $f_\theta : \mathcal{Z} \to \mathbb{R}$. They induce the additive queries

$$F_\theta(D) = \sum_{i=1}^{N} f_\theta(z_i), \qquad \theta \in \Theta. \tag{16}$$

Rather than ask whether individual data values repeat, ask how many independent *functions of one datum* are actually needed to express every query in the family.

Definition 1 (Exact summary rank)**.** Let $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$ be a family of real-valued functions on the data domain. Its *exact summary rank* is

$$r(\mathcal{F}) = \dim \operatorname{span} \mathcal{F}, \tag{17}$$

when this dimension is finite.

Writing $r = r(\mathcal{F})$, choose a basis $\phi_1, \ldots, \phi_r$ for $\operatorname{span} \mathcal{F}$. Every query function has a representation

$$f_\theta(z) = \sum_{j=1}^{r} c_j(\theta)\phi_j(z). \tag{18}$$

Define the r additive summaries

$$M_j(D) = \sum_{i=1}^{N} \phi_j(z_i), \qquad j = 1, \ldots, r. \tag{19}$$

Then

$$F_\theta(D) = \sum_{j=1}^{r} c_j(\theta) M_j(D). \tag{20}$$

Thus every query factors through the map

$$D \longmapsto M(D) = (M_1(D), \ldots, M_r(D)) \longmapsto F_\theta(D). \tag{21}$$

The raw input may contain N or many more scalar coordinates, while the query family sees it only through an r-dimensional exact summary.

4.1 Minimality in the linear-additive model

Without restrictions, "minimum summary dimension" is ill-posed: arbitrary encodings can hide enormous objects inside a single real number. The meaningful claim is therefore made inside a specified representation class.

Proposition 1 (Minimal exact additive summary)**.** *Let $r = \dim \operatorname{span} \mathcal{F} < \infty$. Consider representations using h scalar summaries, with per-observation functions ψ_j and query-dependent recovery coefficients $\alpha_j(\theta)$, of the form*

$$M_j(D) = \sum_i \psi_j(z_i), \quad j = 1, \ldots, h, \qquad F_\theta(D) = \sum_{j=1}^{h} \alpha_j(\theta) M_j(D), \tag{22}$$

Among such representations valid for every D and θ, the minimum number of scalar summaries is exactly

$$h_{\min} = r. \tag{23}$$

Proof. The basis construction above attains $h = r$. Conversely, suppose a representation with h summaries exists. Apply it to a one-point dataset $D = (z)$. Then

$$f_\theta(z) = \sum_{j=1}^{h} \alpha_j(\theta)\psi_j(z),$$

so every f_θ lies in $\operatorname{span}\{\psi_1, \ldots, \psi_h\}$. Therefore $r = \dim \operatorname{span} \mathcal{F} \le h$. Hence $h_{\min} = r$. □

The theorem is elementary but exact: r is the number of additive scalar channels through which the query family can be compressed. It identifies the smallest representation, not yet the cheapest algorithm for reaching it. More broadly, using algebraic structure to replace an apparent operation count by a smaller intrinsic one is a classical theme in arithmetic complexity [2].

For the motivating family

$$f_{a,b}(x, y) = ax + by, \tag{24}$$

the symbols x and y in $\operatorname{span}\{x, y\}$ denote the two coordinate functions. Hence $\operatorname{span} \mathcal{F} = \operatorname{span}\{x, y\}$, $r = 2$, and the summaries $X = \sum x_i$ and $Y = \sum y_i$ are minimal in the stated class.

Takeaway. Exact summary rank counts the fewest additive channels needed to answer the whole query family.

5 Cost-aware redundancy

Rank alone is not runtime. A summary must be cheap to construct, reuse, and reach from the representation actually supplied. Thus the representation size r must be kept distinct from its construction and evaluation costs. A small rank with extremely expensive basis functions may be algorithmically inferior to a larger but cheaper representation.

For a basis $\Phi = (\phi_1, \ldots, \phi_r)$, write schematically

$$T_\Phi(N, Q) = N\, C_{\text{update}}(\Phi) + Q\, C_{\text{query}}(\Phi), \tag{25}$$

Here $T_\Phi(N, Q)$ is total work for N observations and Q queries, $C_{\text{update}}(\Phi)$ is the cost of incorporating one observation into all summaries, and $C_{\text{query}}(\Phi)$ is the cost of answering one query from them. The natural optimization problem is therefore not simply $\min r$, but

$$\min_{\Phi:\ \text{exact recovery}} T_\Phi(N, Q). \tag{26}$$

This is the cost-aware form of exploitable internal redundancy: find a quotient of the computation that preserves all requested answers while minimizing total work.

5.1 Repeated queries

The advantage becomes especially clear when the data are fixed and the parameters vary. For Q weight pairs (a_q, b_q), indexed by $q = 1, \ldots, Q$, direct evaluation costs

$$T^{(Q)}_{\text{direct}} = Q\big[2Nc_\times + (2N - 1)c_+\big]. \tag{27}$$

After computing X, Y once,

$$T^{(Q)}_{\text{summary}} = (2N - 2)c_+ + Q(2c_\times + c_+). \tag{28}$$

The saving is

$$\Delta T = 2(N - 1)\big[Qc_\times + (Q - 1)c_+\big]. \tag{29}$$

The N-dependence is paid once; each new query costs only $O(r)$. This is the regime in which summary dimension starts to behave like a genuine algorithmic resource rather than a cosmetic reformulation. Database systems exploit the same broad tradeoff when selecting materialized aggregate views: pay storage and construction costs once to reduce later query time [3].

Takeaway. The best summary minimizes total construction and query cost, not necessarily dimension alone.

6 Examples: summaries as computational coordinates

Several familiar constructions fit immediately. Polynomial queries of degree at most m have $r = m + 1$, with summaries

$$N, \sum_i x_i, \sum_i x_i^2, \ldots, \sum_i x_i^m. \tag{30}$$

For categorical queries, indicator functions give category counts; for finite Fourier families, the summaries are the selected Fourier coefficients. More generally, if

$$f_\theta(z) = \langle c(\theta), \phi(z) \rangle, \qquad \phi(z) \in \mathbb{R}^r, \tag{31}$$

where $c(\theta) \in \mathbb{R}^r$ is a coefficient vector and $\langle \cdot, \cdot \rangle$ is the Euclidean inner product, then the dataset enters only through $\sum_i \phi(z_i)$. Means, moments, counts, and Fourier coefficients are sufficient here because of the query family, not because a probability model has been assumed.

These examples compress repeated aggregation. The next step asks whether structure among the coordinates of a single input can remove the ambient dimension itself.

Takeaway. Familiar aggregates are coordinates of the smaller space through which a query family sees the data.

7 Input-side structure: beyond a constant factor

The preceding reductions exploit structure in the *query family*. A weighted predictor reveals a complementary possibility: the admissible inputs themselves may occupy fewer computational coordinates. For a feature vector $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and weight vector $w = (w_1, \ldots, w_d) \in \mathbb{R}^d$, consider

$$p_w(x) = w^\top x = \sum_{j=1}^{d} w_j x_j. \tag{32}$$

Pairwise statistical correlation is not enough for an exact reduction. The relevant exact condition is that the inputs lie in a lower-dimensional subspace and that coordinates in that subspace are cheaply accessible. A single known subspace is the simplest case of richer union-of-subspaces signal models [4].

Proposition 2 (Exact input-subspace acceleration)**.** *Let every admissible input satisfy $x = Az$, where $A \in \mathbb{R}^{d \times k}$ is a fixed full-column-rank basis matrix, $k \le d$, and the latent coordinate vector $z \in \mathbb{R}^k$ is supplied or maintained with the input. For a fixed w, after precomputing $\widetilde{w} = A^\top w$, the predictor (32) can be evaluated exactly in $\Theta(k)$ online time under the unit-cost arithmetic model. A generic dense evaluation in the original coordinates takes $\Theta(d)$; hence the speedup is $\Theta(d/k)$. In particular, if $k = o(d)$, the improvement is asymptotic, and if $k = O(1)$, online prediction is $O(1)$ instead of $\Theta(d)$.*

Proof. Associativity gives

$$p_w(x) = w^\top Az = (A^\top w)^\top z = \widetilde{w}^\top z.$$

The one-time dense change of weights costs $O(dk)$. Thereafter, evaluation uses k multiplications and $k - 1$ additions, rather than d and $d - 1$, so the unit-cost ratio is $(2d - 1)/(2k - 1) = \Theta(d/k)$. For generic dense $\widetilde{w}$, an exact algorithm must inspect every z_j: if it omitted one with $\widetilde{w}_j \neq 0$, two inputs differing only in that coordinate would be indistinguishable but would have different predictions. Hence the online bound is $\Theta(k)$. □

The phrase "supplied or maintained" carries the burden of the result. Recovering z from a fresh dense x can cost at least the original scan. More subtly, even a very small intrinsic answer may be spread across many stored coordinates. We now make that distinction exact.

7.1 The rank–accessibility gap

Let $U \subseteq \mathbb{R}^d$ be the k-dimensional space of admissible inputs and let $W \subseteq \mathbb{R}^d$ be a linear family of predictor weights. Write U^* for the dual space of linear functionals on U. The restricted prediction space is

$$L(U, W) = \{x \mapsto w^\top x : w \in W\} \subseteq U^*, \qquad s(U, W) = \dim L(U, W). \tag{33}$$

The quantity $s = s(U, W)$ is the minimum number of arbitrary linear summaries needed to recover every prediction. To distinguish intrinsic dimension from storage access, let $\pi_j(x) = x_j$ be the j-th coordinate functional; $\pi_j|_U$ denotes its restriction to U. Define

$$\tau(U, W) = \min\{|S| : L(U, W) \subseteq \operatorname{span}\{\pi_j|_U : j \in S\}, \quad S \subseteq \{1, \dots, d\}\}. \tag{34}$$

Thus τ is the minimum number of *fixed raw coordinates* whose values permit exact linear recovery of every predictor in W. This access model is deliberately stricter than compressed sensing, where the algorithm may design arbitrary linear measurements of a structured signal [5].

Proposition 3 (Rank–accessibility gap)**.** *If $s(U, W) > 0$, then*

$$1 \le s(U, W) \le \tau(U, W) \le k. \tag{35}$$

Every gap is possible: for all integers $1 \le s \le \tau \le k \le d$, there exist $U, W \subseteq \mathbb{R}^d$ with $s(U, W) = s$ and $\tau(U, W) = \tau$.

Proof. A basis of $L(U, W)$ gives $s(U, W)$ arbitrary summaries; any sufficient family of summaries must span $L(U, W)$, proving minimality. If a coordinate set S realizes τ, then $L(U, W)$ lies in the span of $|S|$ functionals, so $s \le \tau$.

The restrictions $\pi_1|_U, \dots, \pi_d|_U$ span U^*: otherwise some nonzero $x \in U$ would be annihilated by every coordinate functional, forcing $x = 0$. Since $\dim U^* = k$, some k restricted coordinates form a basis, and $\tau \le k$.

For attainability, fix $1 \le s \le \tau \le k \le d$, let $e_1, \dots, e_d$ be the standard basis of $\mathbb{R}^d$, take $U = \operatorname{span}\{e_1, \dots, e_k\}$, and set $W = \operatorname{span}\{w_1, \dots, w_s\}$, where

$$w_1 = e_1 + \cdots + e_{\tau - s + 1}, \qquad w_i = e_{\tau - s + i} \quad (2 \le i \le s).$$

The restricted functionals $\ell_i(x) = w_i^\top x$ have disjoint nonempty supports, hence are independent and give prediction rank s. The first τ coordinates suffice. Conversely, the coordinate functionals form a basis of U^*, so representing ℓ_1 requires each of coordinates $1, \dots, \tau - s + 1$, and each remaining ℓ_i requires its own distinct coordinate. Every sufficient set therefore contains all first τ coordinates. Thus the accessibility is exactly τ. □

At the extreme, $s = 1$ and $\tau = k$: the answer occupies one abstract channel but exact evaluation must fetch all k accessible coordinates. This is why rank alone is not runtime. Ordinary statistical correlations add a second qualification. If $x = Az + e$, where $e \in \mathbb{R}^d$ is the approximation residual, then

$$|p_w(x) - \widetilde{w}^\top z| = |w^\top e| \le \|w\|_2 \|e\|_2. \tag{36}$$

Here $\|\cdot\|_2$ is the Euclidean norm. This is an accuracy–runtime tradeoff, not an exact complexity claim; classical low-rank approximation supplies the canonical matrix version of such a residual tradeoff [6]. Structure pays only when it is exact enough for the requested answer and cheap enough to reach from the supplied representation.

Takeaway. Low intrinsic dimension accelerates prediction only when the needed coordinates are cheap to access.

8 Relation to existing ideas and the novelty boundary

The broad territory is well populated, and any novelty claim should be modest.

Compiler redundancy elimination. Common subexpression elimination removes repeated calculations when reuse is profitable [1]. Our starting equality case is philosophically close, but the later summary-rank view groups computation by a function-space basis rather than by syntactically repeated expressions.

Statistical and computational sufficiency. Classical sufficiency asks for a reduction that preserves inferential information under a statistical model [7]. Vu's computational sufficiency instead begins with a family of procedures and asks which reductions retain enough information to compute them [8]. The present framework is a restricted, explicitly additive and cost-counted instance of that larger idea.

Streaming and sketches. Streaming algorithms replace full data storage with compact state, often permitting approximation; frequency moments are a canonical example [9]. Modern linear-sketch theory likewise asks which low-dimensional measurements suffice for particular query classes. Our exact summary construction is much simpler: when the one-point query functions span a finite-dimensional space, exact linear-additive compression follows immediately.

Coresets. Coresets seek small data representations that preserve a family of objective values, usually approximately and problem-dependently [10]. Exact summary rank is not a coreset size in general: the summaries need not be weighted input points at all. It is closer to a coordinate representation of the aggregate computation.

Accordingly, the propositions are elementary linear algebra, not deep standalone theorems. The contribution is the *trajectory*: local savings give way to minimal global summaries; summaries yield amortization; accessible input subspaces reduce the order of prediction; and the rank–accessibility theorem finally proves that intrinsic dimension and raw access cost can be separated by a factor of k.

This framing also suggests a broader research question:

> Given a computation family, a supplied representation, and a machine model, what is the cheapest exact or approximate factorization of the requested answers?

The answer depends on dimension and on the cost of discovering, accessing, updating, and evaluating the representation. "Internal redundancy" is therefore relational: it belongs jointly to the data, queries, storage format, and machine model.

Open directions. Approximate summaries could be chosen jointly with an error budget; broader access models could replace fixed coordinates by adaptive probes, blocks, or compressed storage; and automatic discovery could be combined with dynamic maintenance. The same objective can price memory traffic, communication, or energy rather than equal-cost arithmetic.

Takeaway. The framing unifies established ideas by tracking both compression and the cost of reaching it.

9 Conclusion

The results strengthen along a deliberate trajectory. Local equalities give instance-sensitive savings governed by γ, but only after paying detection cost. Global linearity dominates that tactic, halving the motivating arithmetic without testing equality. For additive query families,

$$r(\mathcal{F}) = \dim \operatorname{span} \mathcal{F} \tag{37}$$

is exactly the minimum additive summary dimension, and reuse turns repeated scans into $O(r)$ queries. Exact input dependence can then change the order from $\Theta(d)$ to $\Theta(k)$.

The final result supplies the qualification. If s is the intrinsic prediction rank and τ is the number of fixed raw coordinates needed to expose it, then

$$s \leq \tau \leq k, \tag{38}$$

and every gap occurs. A one-dimensional answer can still require k coordinate reads. Compression becomes computation only when the compressed coordinates are accessible.

The broader principle is simple:

> *Ask whether the computation factors through a smaller space, and count the cost of reaching that space.*

Exact relations, approximate correlations, reuse, and hardware costs are different versions of one problem: find the cheapest answer-preserving factorization from the representation actually in hand.

Takeaway. Structure becomes computationally valuable only when the algorithm can reach a smaller answer-preserving space cheaply.

References

[1] J. Cocke. Global common subexpression elimination. *ACM SIGPLAN Notices*, 5(7):20–24, 1970.

[2] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.

[3] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *Proceedings of ACM SIGMOD*, pages 205–216, 1996.

[4] N. Rao, B. Recht, and R. D. Nowak. Signal recovery in unions of subspaces with applications to compressive imaging. arXiv:1209.3079, 2012.

[5] E. J. Candès and T. Tao. Near-optimal signal recovery from random projections: universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.

[6] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[7] R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society A*, 222:309–368, 1922.

[8] V. Q. Vu. Group invariance and computational sufficiency. arXiv:1807.05985, 2018.

[9] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[10] D. Feldman. Introduction to core-sets: an updated survey. arXiv:2011.09384, 2020.