Certified Candidate Search for Sparse Recursive Boosting

Anonymous

November 10, 2026

Abstract

Joint coefficient refitting improves the loss and sparsity of recursive boosting under a fixed cap on example evaluations. We examine the additional cost of finding the next branch. For Boolean gates, the training incidence relation can be constructed from the active gates on each row, omitting every conjunction with empty training support. A signed mass bound certifies entire prefix families, including when the search stops early. Fixed-point accumulation makes the linear oracle certificate exact for the supplied gradient, with an explicit rounding allowance. We verify the oracle against exhaustive search and compare corrective Newton refitting with pairwise Frank–Wolfe on the same tilted objective and coefficient ball. Sparse indexing yields substantial savings over nominal enumeration. Prefix search reduces the number of incidences examined, with mixed timing results. The complete code, per-run records, and certificate checks accompany the note.

Keywords. Recursive boosting, Boolean conjunctions, safe pruning, tilted risk, conditional gradient.

1 Introduction

The recursive boosting manuscript [1] gives a polynomial bound on the number of degree-bounded candidates, a maximum-entropy error selector, and a finite conditional-gradient fitting procedure. The nominal candidate count can still be large when the gate alphabet has hundreds of elements. Candidate scoring can then dominate an update even if the fitted ensemble has only a few branches.

This companion studies a sparse Boolean specialization. Its resource bound is controlled by the number of row–conjunction incidences. The construction uses each training row's active gates to generate its incident terms. There is no need to test the support of every possible conjunction. A prefix search can then use the sign-separated residual masses of a parent support to bound the correlations of every descendant.

The relevant prior work is extensive. Entropy duality and joint coefficient correction are established in boosting [2]. Fully corrective and pairwise Frank–Wolfe are established polytope optimization methods [3], and conditional gradients can use inexact or lazy subproblem solvers [4]. Safe pattern pruning uses the containment structure of pattern supports to screen families [5]. WHInter develops specialized working sets for high-dimensional sparse second-order interactions [6]. Our contribution is an audited finite incidence specialization for the recursive boosting model, with an integer frontier certificate, an incidence-based cost bound, and matched-objective comparisons.

The earlier experiments measured loss at several caps on training-row evaluations. The larger study measures two other quantities directly. The first is the work of a linear oracle with a fixed input gradient. The second is total fitting time to a prescribed numerical gap, with the index construction charged to each solver. These measurements reveal when candidate reductions translate into shorter running time.

2 Empirical Boolean dictionary

Let $X = \{1, \dots, n\}$ be the training rows, with $n \geq 1$. There are b Boolean gates. The set of active gates on row i is $G_i \subseteq [b]$. For a subset $a \subseteq [b]$ with $|a| \leq k$, define

$$I_a = \{i \in X : a \subseteq G_i\}, \qquad c_a = 1 + |a|, \qquad A_a = \frac{\mathbf{1}_{I_a}}{c_a}.$$

The empty conjunction has support X and cost one. The note uses one constant leaf. The output dimension C can be greater than one.

The feasible prediction set is

$$\mathcal{K}_R = R \operatorname{conv}\big(\{0\} \cup \{\pm A_a e_c^\top : a \in \mathcal{A}_k,\ 1 \leq c \leq C\}\big).$$

Equivalently, $Z = A\Theta$ with total coefficient norm $\|\Theta\|_1 \leq R$. The coefficient costs act in the original coordinates through the normalization c_a. A canonical word for a lists its gate indices in increasing order.

Lemma 2.1 (Reduction of Boolean words)**.** *The empirical prediction hull of the Boolean word dictionary with the cost* $1 +$ *word length equals* $\mathcal{K}_R$*. Deleting atoms with empty training support preserves this hull.*

Proof. Gate multiplication commutes and each Boolean gate is idempotent. If a word of length d has distinct gate set a, its column is $(1 + |a|)/(1 + d)$ times A_a. The factor lies in $[0, 1]$, and the hull contains zero and both signs of A_a. Each distinct conjunction is itself a word. An empty support contributes only zero. This proves both equalities. □

The equality for repeated gates holds on any Boolean input. The deletion of empty supports is an empirical statement. A conjunction absent from training may fire on another sample. Held-out evaluation uses the actual selected conjunctions and the same gate encoding.

Definition 2.2 (Incidence size)**.** Write $s_i = |G_i|$, $M_{\mathrm{nom}} = \sum_{d=0}^{k} \binom{b}{d}$, and $M_{\mathrm{nz}} = |\{a : I_a \neq \varnothing\}|$. We use $\binom{u}{v} = 0$ for $v > u$. Let

$$P = \sum_a |I_a| = \sum_{i=1}^{n} \sum_{d=0}^{\min(k,s_i)} \binom{s_i}{d}.$$

Theorem 2.3 (Incidence construction)**.** *Given sorted sparse row supports G_i, the nonempty conjunctions and all their row postings can be generated using P incidence insertions. A hash-based implementation has expected construction work*

$$O\big(\mathrm{nnz}(G) + kP + kM_{\mathrm{nz}} \log M_{\mathrm{nz}} + n\big),$$

allowing $O(k)$ work for each key. The resulting index uses $O(n + P + kM_{\mathrm{nz}})$ storage. A direct indexed oracle uses $O(CP + CM_{\mathrm{nz}})$ arithmetic per gradient.

Proof. For each row i, emit (a, i) for each subset $a \subseteq G_i$ of size at most k. The emission condition is exactly $i \in I_a$. Grouping these emissions by their increasing gate tuple produces every nonempty posting list and no empty one. Sorting the distinct tuple labels costs $O(kM_{\mathrm{nz}} \log M_{\mathrm{nz}})$. The storage bound follows from the postings, keys, and sparse row offsets. To score all supported terms, accumulate each C-component residual vector into the columns incident to its row. Normalize by c_a and take the largest absolute component. This performs CP component additions and CM_{nz} score comparisons. □

A deterministic alternative sorts all emitted pairs before grouping, replacing the expected hash work with $O(kP \log P)$ comparison work. Candidate labels take $O(\log b)$ bits per gate index. The work bound treats an index as one word.

Corollary 2.4. *If $s_i \leq s$ for all i, then $P \leq n \sum_{d=0}^{k} \binom{s}{d}$. For fixed s and k, the incidence scan is linear in nC, while the nominal dictionary grows as b^k for fixed positive k.*

This bound requires sparsity within each training row. If every gate is active on every row, $P = nM_{\mathrm{nom}}$. The code stores the index in compressed sparse column form. The Newton refit materializes only the admitted columns.

3 Certified search over prefixes

Let $Q \in \mathbb{Z}^{n \times C}$ be an integer residual array. An exact signed correlation score is

$$r_Q(a) = \max_c \frac{\left|\sum_{i \in I_a} Q_{ic}\right|}{c_a}.$$

The largest absolute score identifies a minimizing signed vertex of the linear oracle. An intercept and any cached incumbent candidates may be scored before search.

Theorem 3.1 (Signed mass bound)**.** *For a prefix p of depth d, define*

$$P_{pc} = \sum_{i \in I_p} [Q_{ic}]_+, \qquad N_{pc} = \sum_{i \in I_p} [-Q_{ic}]_+,$$

$$U_p = \frac{\max_c \max(P_{pc}, N_{pc})}{d + 2}.$$

Every proper descendant a of p satisfies $r_Q(a) \leq U_p$.

Proof. A descendant adds gates, hence $I_a \subseteq I_p$. Its signed numerator in coordinate c lies in $[-N_{pc}, P_{pc}]$, and its cost is at least $d+2$. Taking the maximum gives the bound. □

For a fixed coordinate, the larger of the positive and negative masses is the largest absolute sum over arbitrary subsets of I_p. A subset containing all entries of the corresponding sign attains it. The bound thus uses all the information provided by support inclusion alone. Tighter bounds can use the constraints imposed by the remaining gates.

3.1 Frontier and stopping certificate

The search scores a prefix before adding its family of proper descendants to a priority queue. Each family is keyed by its bound from Theorem 3.1. Expanding a family scores its immediate children and replaces the family by their proper-descendant families. The code accumulates the signed masses of siblings in sparse matrix batches. Cached scores keep each posting list from being scored twice during one search.

Theorem 3.2 (Anytime linear oracle)**.** *Let B be the best exact score seen. Let U be the larger of B and the maximum pending family bound, taking $U = B$ for an empty queue. Then*

$$B \;\leq\; \max_{a \in \mathcal{A}_k} r_Q(a) \;\leq\; U.$$

The inequality remains valid when the algorithm stops after a budgeted number of prefix expansions.

Proof. Each supported, unscored conjunction is in an unexpanded family or a discarded family. Expanding a prefix family preserves this covering relation by its child decomposition. A family is discarded only when its bound is at most the incumbent at that time. The incumbent is nondecreasing. Empty-support atoms have score zero, which the hull permits through its zero vertex. Thus every candidate is bounded by U, and the incumbent is attained by a scored candidate. □

If $U = B$, the best score is exact for Q, including ties. When a budget is exhausted first, the difference $U - B$ is an explicit loss in linear-oracle accuracy. The number

of scored postings is at most P for one prefix search. Maintaining the queue adds comparison and bookkeeping work. In particular, a reduction in postings read need not reduce elapsed time.

3.2 Finite precision

The implementation accepts a finite binary64 array q, scales it by 2^s, and rounds to integer Q. The range guard ensures that multiplying by the power of two is exact in the supported range and that every accumulator fits in signed 64 bits. Scores and queue bounds compare rational numbers exactly. For every component, $|q_{ic} - 2^{-s}Q_{ic}| \le 2^{-s-1}$.

Corollary 3.3 (Selection error in supplied units)**.** *With $E = n/2^{s+1}$, the score of the selected signed atom for q is at least $2^{-s}B - E$, and the optimal score is at most $2^{-s}U + E$. The additive error of the minimization oracle on $\mathcal{K}_R$ is at most*

$$R\big(2^{-s}(U - B) + 2E\big).$$

Proof. For any fixed signed column and output coordinate, the rounded score differs by at most $|I_a|2^{-s-1}/c_a \le E$. Apply this to the returned atom and to every possible maximizer, then scale by the vertex radius. □

The certificate is exact for the supplied gradient numbers. The computation of a nonlinear loss gradient is a separate source of numerical error. If q approximates the exact derivative g with a known bound on $\|q - g\|_1$, the additional error in a difference of two feasible linear scores is at most $2R\|q - g\|_1$.

3.3 Finite incidence semantics

The prefix order is a category. The support assignment is contravariant, since $p \preceq a$ gives an inclusion $I_a \hookrightarrow I_p$. Let $E_{\mathrm{inc}} = \coprod_a I_a$, with projections $r : E_{\mathrm{inc}} \to X$ and $c : E_{\mathrm{inc}} \to \mathcal{A}_k$. Correlation numerators are obtained by pulling the residual field back along r and summing along the fibers of c. The construction is a finite span interpretation of evaluation, compatible with the polynomial presentation in [1].

For a finite map $f : U \to V$, write $f_!w(v) = \sum_{f(u)=v} w(u)$. The composition law for $f_!$ follows by regrouping finite sums. For a pullback of f, each new fiber is identified with the corresponding old fiber, so pullback and fiberwise sum obey the finite Beck–Chevalley identity. In particular, splitting the rows into disjoint blocks permits positive and negative masses to be computed separately and added before taking their maximum. This identifies precisely which certificate data combine across disjoint samples.

4 Fitting and the gap diagnostic

We use the tilted empirical risk from [7].

$$F_\tau(Z) = \tau \log\left(\frac{1}{n}\sum_i \exp\left(\frac{\ell_i(Z_i)}{\tau}\right)\right).$$

Its gradient in the unnormalized Euclidean pairing on predictions is $g_i = p_i \nabla \ell_i(Z_i)$, where p_i is the Gibbs distribution of the example losses. This distribution maximizes expected loss minus τ times relative entropy to the uniform distribution. We use $\tau = 1$.

For binary logistic loss and margin m_i, the resulting objective is $\log(1 + n^{-1}\sum_i e^{-m_i})$. The minimizers on the coefficient ball therefore coincide with those of empirical exponential margin loss. The implementation uses the tilted loss as its reporting and stopping objective.

Proposition 4.1 (Gap with an inexact gradient)**.** *For feasible Z, use a supplied gradient q and a frontier certificate U for its fixed-point encoding. Then*

$$F_\tau(Z) - \min_{W \in \mathcal{K}_R} F_\tau(W) \le \langle q, Z\rangle + R(2^{-s}U + E) + 2R\|g - q\|_1,$$

where $g = \nabla F_\tau(Z)$.

Proof. By convexity, suboptimality is at most $\sup_{W \in \mathcal{K}_R}\langle g, Z - W\rangle$. Every coordinate of a feasible prediction lies in $[-R, R]$. Substituting q introduces error at most $2R\|g - q\|_1$. The remaining supremum is the current pairing plus R times the maximum signed correlation for q, bounded by Corollary 3.3. □

The reported experiments use floating-point gradients and current pairings. Their gaps are numerical optimization diagnostics. The exact integer verification concerns the linear oracle and its rounding allowance.

Corrective fitting. An update admits the best signed candidate and applies a scheduled conditional-gradient step as a safeguard. It then takes damped Newton steps over the coefficients of the admitted columns, with the original total ℓ^1 radius and monotone backtracking. It adds at most one new branch each round. The scanned and prefix versions use the same correction. The prefix version allows twelve family expansions before falling back to the complete indexed scan if necessary.

The usual safeguarded convergence argument applies. If F is L-smooth in the empirical prediction norm, the feasible diameter is at most $2R$. With per-round linear-oracle error ε_t, $\gamma_t = 2/(t + 2)$, and $h_t = F(Z_t) - F^*$, the scheduled step gives

$$h_{t+1} \le (1 - \gamma_t)h_t + \gamma_t\varepsilon_t + 2LR^2\gamma_t^2.$$

Any accepted correction preserves this inequality. A uniform upper bound $\bar{\varepsilon}$ on the oracle errors gives $h_T \le 8LR^2/(T + 2) + \bar{\varepsilon}$ for $T \ge 1$ by induction. The bound

Task	Corrective	SGD-1	SGD-32
Linear	0.011164	0.012122	0.14507
XOR	1.1521e-20	2.3837e-09	0.16043
Two moons	0.37857	0.38818	0.38477
Breast cancer	0.24573	0.24815	0.30222
Iris	0.24756	0.29126	0.41478
Digits	1.543	1.555	1.5785
Sparse interactions	0.0012191	0.48599	0.96919

Table 1: Mean held-out tilted risk under the 50-pass cap. Each entry uses five paired splits.

counts outer updates, and measured fitting time includes the internal loss calls, sparse operations, and refit arithmetic.

Pairwise comparison. Pairwise Frank–Wolfe [3] maintains nonnegative weights on signed vertices and zero. At a round it moves mass from an active vertex of maximal linear score to the current minimizing vertex. We initialize the step with the directional curvature of F_τ, cap it by the donor mass, and use monotone backtracking. It uses the same indexed scan, objective, coefficient radius, and evaluation cap as the corrective solver.

5 Experiments

5.1 Evaluation budgets against SGD

The previous joint-refitting experiment uses six standard tasks and a sparse interaction regression, with five paired splits per task. The degree-two dictionary, temperature, and coefficient radius are shared within each task. Validation chooses a checkpoint and configuration. The two SGD baselines use compiled single-threaded updates with batch sizes one and 32.

One pass charges n training-example loss or gradient evaluations. Joint loss and gradient calculation counts once. Extra loss evaluations for line search also count. This cap applies to each candidate fitting path. Separate validation and tuning costs are in the archived records.

Table 1 reports mean held-out tilted risk under the cap of 50 passes. The corrective solver has the lowest measured mean on each task and wins against both baselines on 28 of 35 paired splits, using a tolerance of 10^{-10} for numerical ties. On XOR, corrective fitting and single-example SGD both approach zero. The sparse interaction generator deliberately uses a target supported on a few degree-two terms.

On Digits, the corrective model averages 10.4 branches and batch-32 SGD averages 39.6. The corresponding fitting prefixes take 146 and 58 milliseconds. On the sparse interaction task, the fitting prefixes average 11, 116, and 87 milliseconds for corrective fitting, SGD, and batch-32 SGD. These prefix times exclude tuning. The rankings vary with the evaluation budget. Single-example SGD leads at 20 passes, and also on Iris at 200 passes. The archive retains all four caps.

Case	Nominal	Supported	Incidences
Sparse 64	43,745	12,882	76,800
Sparse 128	349,633	20,788	76,800
Sparse 512	22,370,049	33,663	76,800
Digits	43,745	7,806	149,017

Table 2: Degree-three dictionary sizes and index incidence counts, medians over three seeds.

Case	Nominal	Indexed	Prefix	Reads
Sparse 64	540.25	0.30	0.37	9,600
Sparse 128	–	0.26	0.49	9,600
Sparse 512	–	0.33	0.98	9,600
Digits	68.54	0.30	0.26	4,089

Table 3: Median query milliseconds at the fitted intercept. Reads counts the postings examined by prefix search. The nominal scan was timed on the 64-gate dictionaries.

5.2 Degree-three oracle comparisons

The new sparse problems use 1,200 training rows, 400 test rows, and $b = 64, 128, 512$ gates. Four signal variables each activate one of two indicators. Three additional gates per row are drawn without replacement from the background alphabet. Thus $s_i = 7$ and $P = 1200(1+7+21+35) = 76{,}800$. The square-loss target combines two planted pairwise interactions, a main effect, and independent Gaussian noise of standard deviation 0.05. The coefficient radius is 12.

The real-data comparison uses the 3-versus-8 subset of Digits. A gate fires at a pixel intensity of at least 12. There are 267 training and 90 test rows in each stratified split. The loss is binary logistic, and the radius is 36. Both experiments use degree three, temperature one, and seeds 101, 211, and 307.

The timed queries use the gradient at zero and at a fitted intercept. For regression the intercept solves the one-dimensional derivative equation. For binary classification it is one half the log class-count ratio. Table 3 shows the fitted-intercept query, which avoids a large initial intercept score on the sparse synthetic tasks. All zero-state results are in the archive. The nominal oracle streams every degree-bounded Boolean conjunction in blocks, including zero supports. The indexed and prefix oracles use the identical fixed-point gradient and exact rational score comparisons.

All complete prefix scores equal the complete indexed score. On the two 64-gate settings the independently enumerated nominal oracle gives the same score. The 512-gate dictionary has 22,370,049 nominal terms and a median of 33,663 supported terms. At the fitted intercept, prefix search scores 513 terms and reads 9,600 postings, compared with 76,800 for the full indexed scan. The prefix query takes about one millisecond and the indexed query takes about a third of a millisecond.

Index construction has a median cost of 84 milliseconds on the sparse 64-gate task. The subsequent complete indexed scan takes about 0.30 milliseconds, and

Case	Solver	ms	Passes	Hit
Sparse 64	Corrective scan	105	68	3/3
	Corrective prefix	106	68	3/3
	Pairwise	155	157	3/3
Sparse 512	Corrective scan	235	68	3/3
	Corrective prefix	254	68	3/3
	Pairwise	286	157	3/3
Digits	Corrective scan	447	268	2/3
	Corrective prefix	479	268	2/3
	Pairwise	199	265	3/3

Table 4: Median total time including index construction, median example-evaluation passes, and number of runs reaching gap 0.02 within the cap of 300 passes.

the streamed nominal scan takes about 540 milliseconds. These measurements include the cost of constructing the support index separately from a gradient query. The index is then available for every update of a fitting run.

5.3 Matched gap fitting

The solver comparison stops at numerical gap 0.02 or at 300 full-sample loss or gradient evaluations. Its wall time includes construction of the same support index for each method. All solvers use the same fixed radius and temperature on a given split. The search and refit settings are fixed across the splits. Each table entry is a median over three seeds, and target attainment is counted separately.

On both synthetic sizes, scanned corrective fitting reaches the gap target on all three seeds with fewer evaluations and shorter total time than pairwise Frank–Wolfe. Its median active support is seven branches, compared with eleven for the pairwise method. On Digits, pairwise Frank–Wolfe reaches the target on all three seeds and takes less time. Both corrective variants reach it on two seeds. The timing median includes a capped run for each corrective variant on that task.

On sparse 512, the prefix version reads a median of 189,953 postings over a fitting run. The complete indexed version reads 614,400. Total fitting time rises from roughly 235 to 254 milliseconds. This is a reduction in candidate work with a modest timing penalty in the measured implementation. The measurements distinguish the benefit of sparse incidence indexing from the additional effect of ordering the supported candidates by certified bounds.

6 Reproduction and scope

The archive includes the benchmark script, the raw oracle and solver records, and the trajectories used to calculate the tables. The integer verifier compares with nominal enumeration and exact rational evaluation of the supplied binary64 gradients. It tests 42 Boolean dictionaries and four early stopping budgets, including scalar and multiclass residuals, all-zero and dense gates, and a degree cap above the gate count.

Run the commands below from the `recursive-boosting` directory with the pinned Python dependencies in `requirements.txt`.

```
python certified/verify.py
python certified/benchmark.py
python certified/make_tables.py
```

The prior benchmark, including the compiled SGD baselines, is supplied in `followup`. All experimental datasets are generated from recorded seeds or loaded from the public scikit-learn Digits dataset [8].

The strongest result is the incidence bound and the exact omission of empty supports. The prefix certificate gives a useful accounting of unresolved candidate families. In the tested implementation, the complete sparse scan is often faster despite reading more postings. The experiments support a choice between sparse scan and prefix search based on the measured incidence distribution and the cost of queue operations.

Authorship and automation. An anonymous human submitter commissioned this companion. An OpenAI assistant derived and checked the arguments, wrote the code, and executed the reported comparisons. The journal record for the original manuscript supplies its editorial status.

References

[1] Anonymous. A higher categorical theory of polynomial-branch recursive boosting with maximum-entropy error selection. *Higher Categorical Semantics*, 2026. Published article.

[2] C. Shen and H. Li. On the dual formulation of boosting algorithms. *IEEE TPAMI*, 32(12), 2216–2231, 2010. doi:10.1109/TPAMI.2010.47.

[3] S. Lacoste-Julien and M. Jaggi. On the global linear convergence of Frank–Wolfe optimization variants. *NeurIPS*, 2015. https://arxiv.org/abs/1511.05932.

[4] G. Braun, S. Pokutta, D. Tu, and S. Wright. Blended conditional gradients. *PMLR* 97, 735–743, 2019. https://proceedings.mlr.press/v97/braun19a.html.

[5] K. Nakagawa, S. Suzumura, M. Karasuyama, K. Tsuda, and I. Takeuchi. Safe pattern pruning: An efficient approach for predictive pattern mining. *KDD*, 2016. https://arxiv.org/abs/1602.04548.

[6] M. Le Morvan and J.-P. Vert. WHInter: A working set algorithm for high-dimensional sparse second order interaction models. *PMLR* 80, 3635–3644, 2018. https://proceedings.mlr.press/v80/morvan18a.html.

[7] T. Li, A. Beirami, M. Sanjabi, and V. Smith. On tilted losses in machine learning: Theory and applications. *JMLR* 24(142), 1–79, 2023. https://jmlr.org/papers/v24/21-1095.html.

[8] Scikit-learn developers. Optical recognition of handwritten digits dataset and `load_digits` documentation. https://scikit-learn.org/stable/datasets/toy_dataset.html.