LightGBM: A Highly Efficient Gradient Boosting Decision Tree

How two novel techniques, Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), accelerate GBDT training by over 20x with no loss in accuracy.

Based on the work by Guolin Ke, Qi Meng, Thomas Finley, et al. (Microsoft Research & Peking University). NIPS 2017.

GBDT is a dominant force in machine learning.

Gradient Boosting Decision Tree (GBDT) is a widely-used algorithm, renowned for its efficiency, accuracy, and interpretability. It achieves state-of-the-art performance in numerous tasks, including classification, click prediction, and learning to rank. Effective implementations like XGBoost and pGBRT have become standard tools for data scientists.

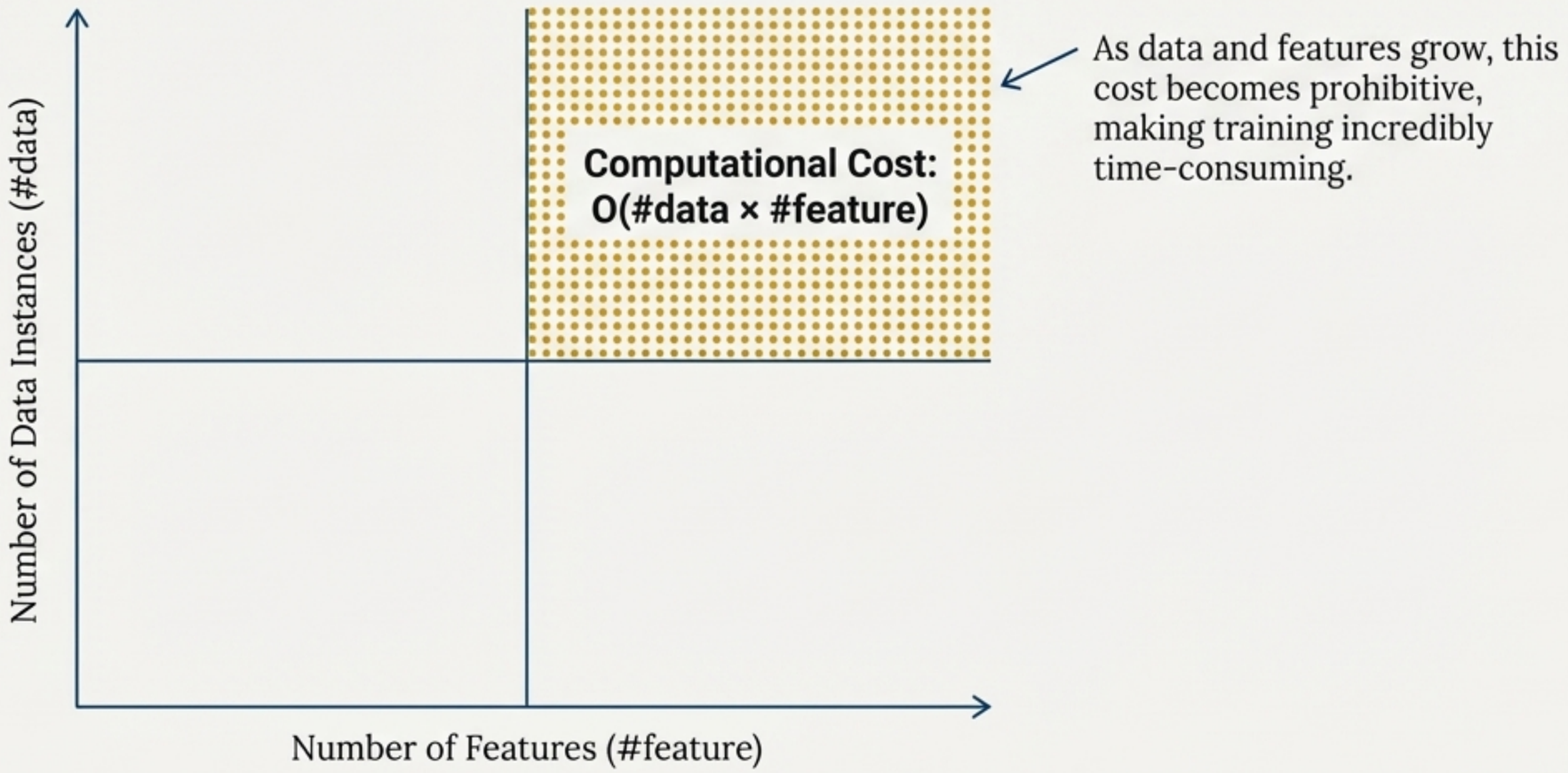
Accuracy

Efficiency

Interpretability

But GBDT faces a computational wall with big data.

The primary challenge is the trade-off between accuracy and efficiency on large datasets. Conventional GBDT implementations must scan every data instance for every feature to find the best split points. The core time-consuming operation is building feature histograms, which dominates the computational complexity.



LightGBM: Breaking the Accuracy vs. Efficiency Trade-off.

LightGBM is a new GBDT implementation designed specifically for speed and efficiency on large-scale data.

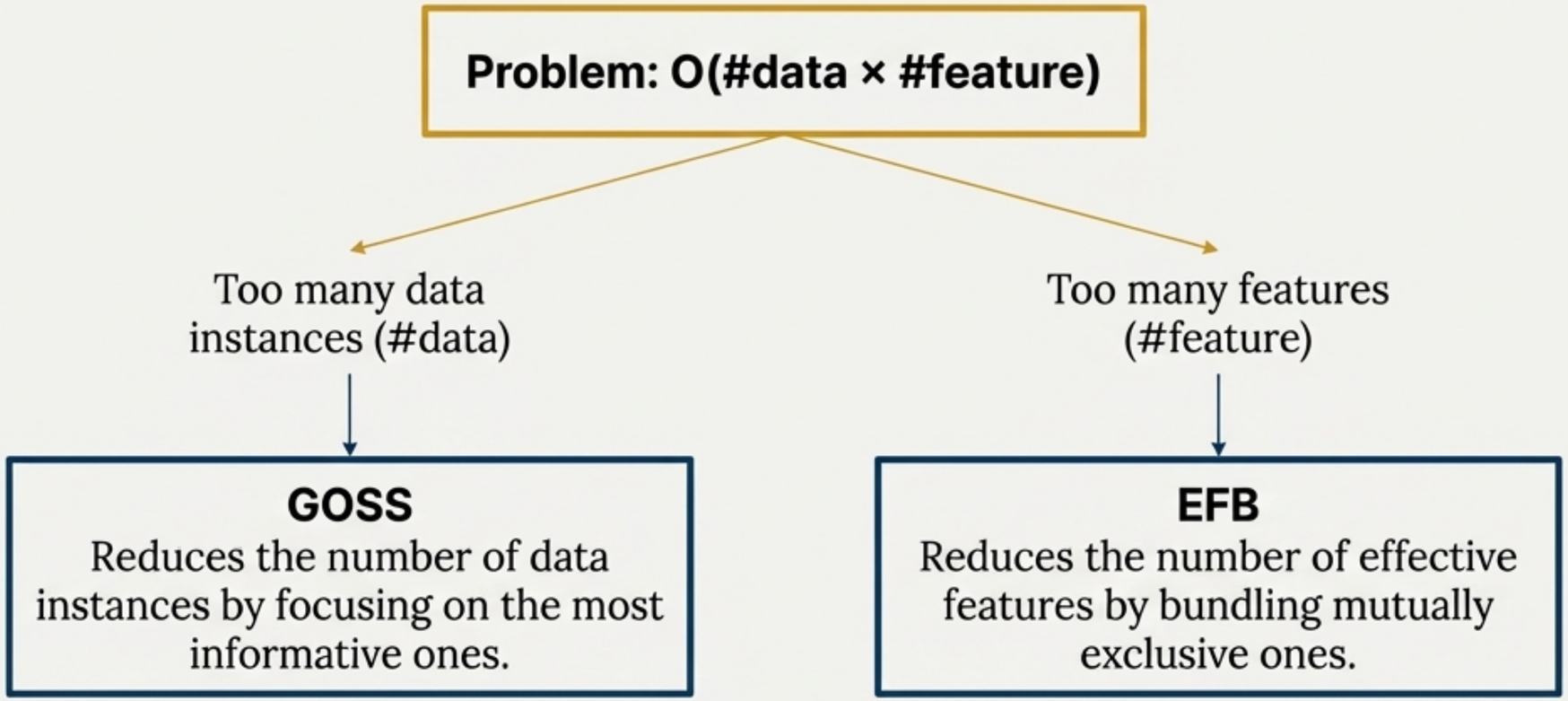
It tackles the computational bottleneck by fundamentally rethinking how data and features are handled during training.

The core of its performance comes from two novel techniques:

1. Gradient-based One-Side Sampling (GOSS)

2. Exclusive Feature Bundling (EFB)

Two targeted techniques for the two primary bottlenecks.



By intelligently reducing both #data and #feature, LightGBM can substantially speed up the training of GBDT.

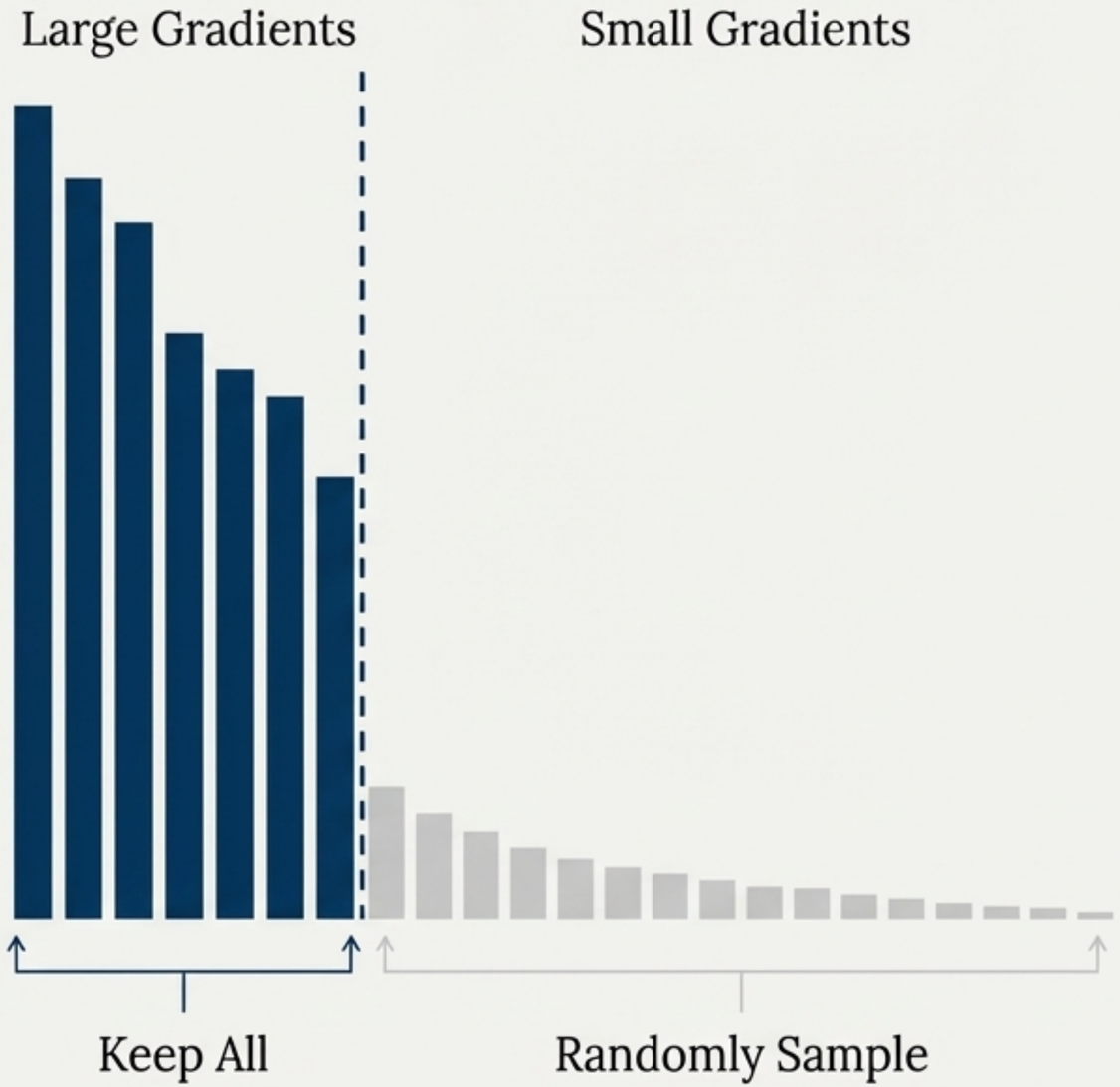
GOSS: Focus on the "under-trained" instances.

In GBDT, data instances with larger gradients contribute more to the information gain. These are “under-trained” instances where the model is still making significant errors.

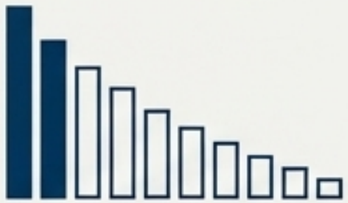
The Idea:

Instead of uniform random sampling (which can hurt accuracy), we can achieve a more accurate gain estimation by:

1. Keeping all instances with large gradients.
2. Performing random sampling only on the instances with small gradients.



How Gradient-based One-Side Sampling works.



Step 1: Sort & Select

Sort all data instances by the absolute value of their gradients in descending order.

Select the top `a`% of instances (the large gradient set).

Step 2: Sample

From the remaining (1 - `a`)% of instances, randomly sample `b`% (the small gradient set).

Step 3: Re-weight & Train

Combine the two sets. Amplify the gradients of the small gradient set by a constant factor `(1-a)/b` to compensate for the sampling.

Train the next tree using this smaller, re-weighted dataset.

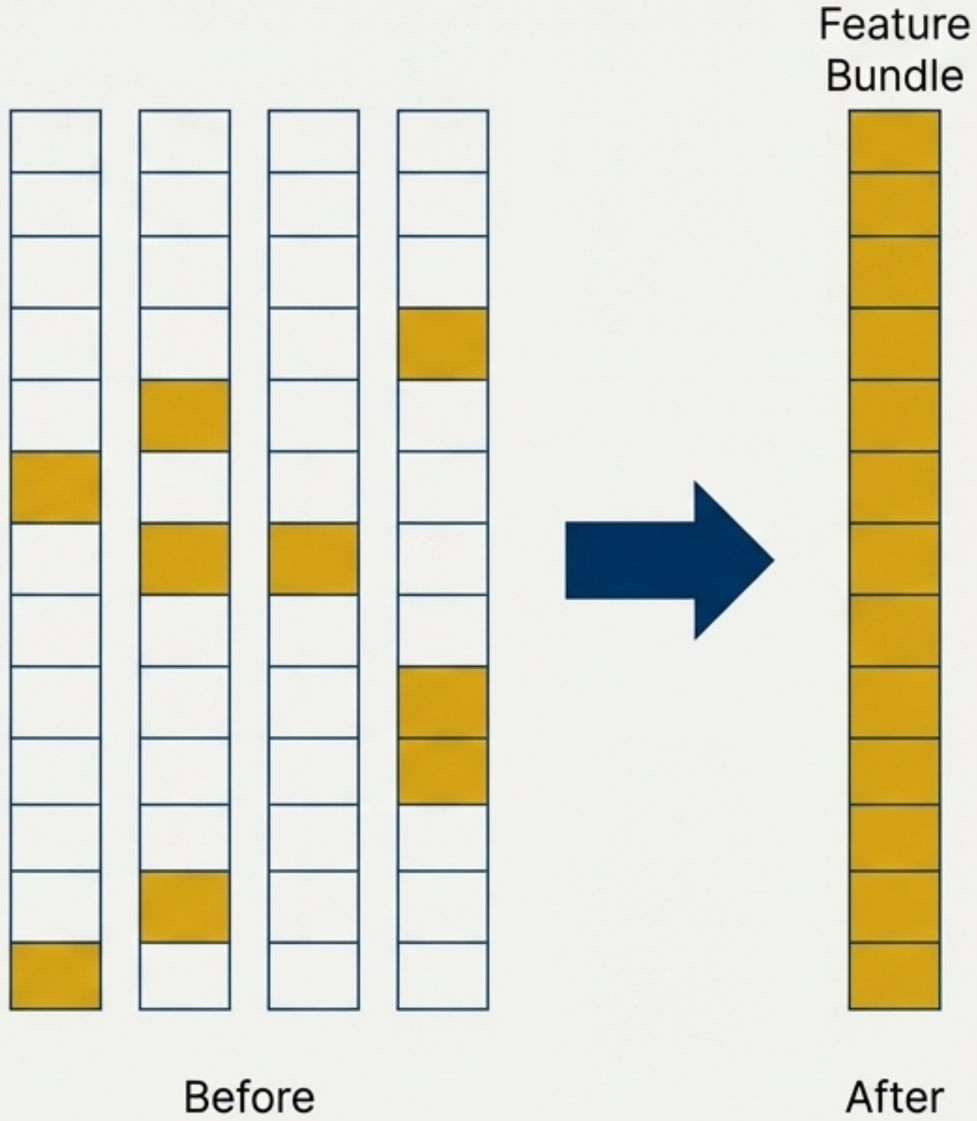
GOSS provides an accurate estimate of the information gain using a much smaller dataset, leading to a significant speed-up.

EFB: Bundling features that rarely conflict.

High-dimensional feature spaces are often very sparse. Many features are “mutually exclusive,” meaning they rarely take non-zero values simultaneously.

One-hot encoded categorical features (e.g., a data point can be from `country:UK` or `country:France`, but not both at the same time).

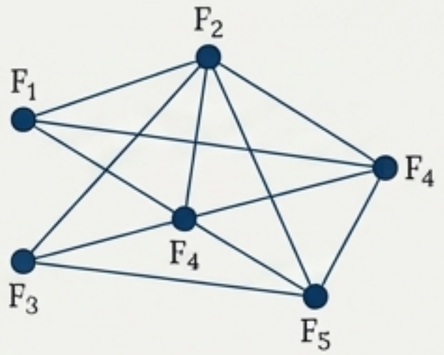
We can design a “nearly lossless” approach by safely bundling these exclusive features into a single, denser feature. This reduces the effective number of features the algorithm needs to process.



How Exclusive Feature Bundling works.

Model as a Graph Problem

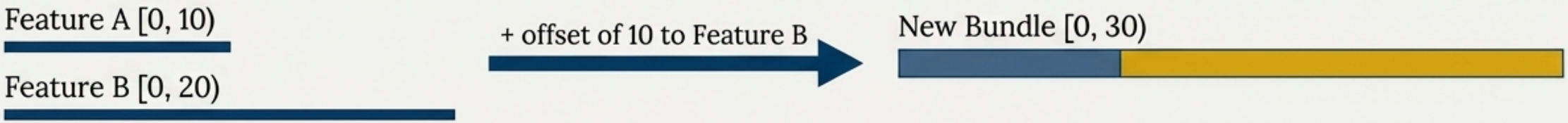
Finding the optimal bundling is NP-hard. It can be modelled as a graph colouring problem where features are vertices and an edge exists between any two non-exclusive features.



Greedy Bundling Algorithm

A greedy algorithm sorts features (e.g., by degree or non-zero count) and iterates through them, either adding a feature to an existing bundle (if conflict is low) or creating a new one.

Constructing the Bundle



Within a bundle, original feature values are preserved by adding offsets.

The complexity of histogram building is reduced from $O(\#data \times \#feature)$ to $O(\#data \times \#bundle)$, where #bundle << #feature.

Putting LightGBM to the test against the best.

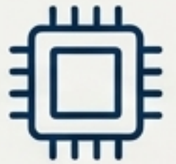
Baselines

- `xgb_exa`: XGBoost with pre-sorted algorithm.
- `xgb_his`: XGBoost with histogram-based algorithm.
- `lgb_baseline`: LightGBM with GOSS and EFB disabled.

Datasets

Name	#data	#feature
Allstate Insurance Claim	12M	4k
Flight Delay	10M	700
LETOR	2M	136
KDD10	19M	29M
KDD12	119M	54M

Environment

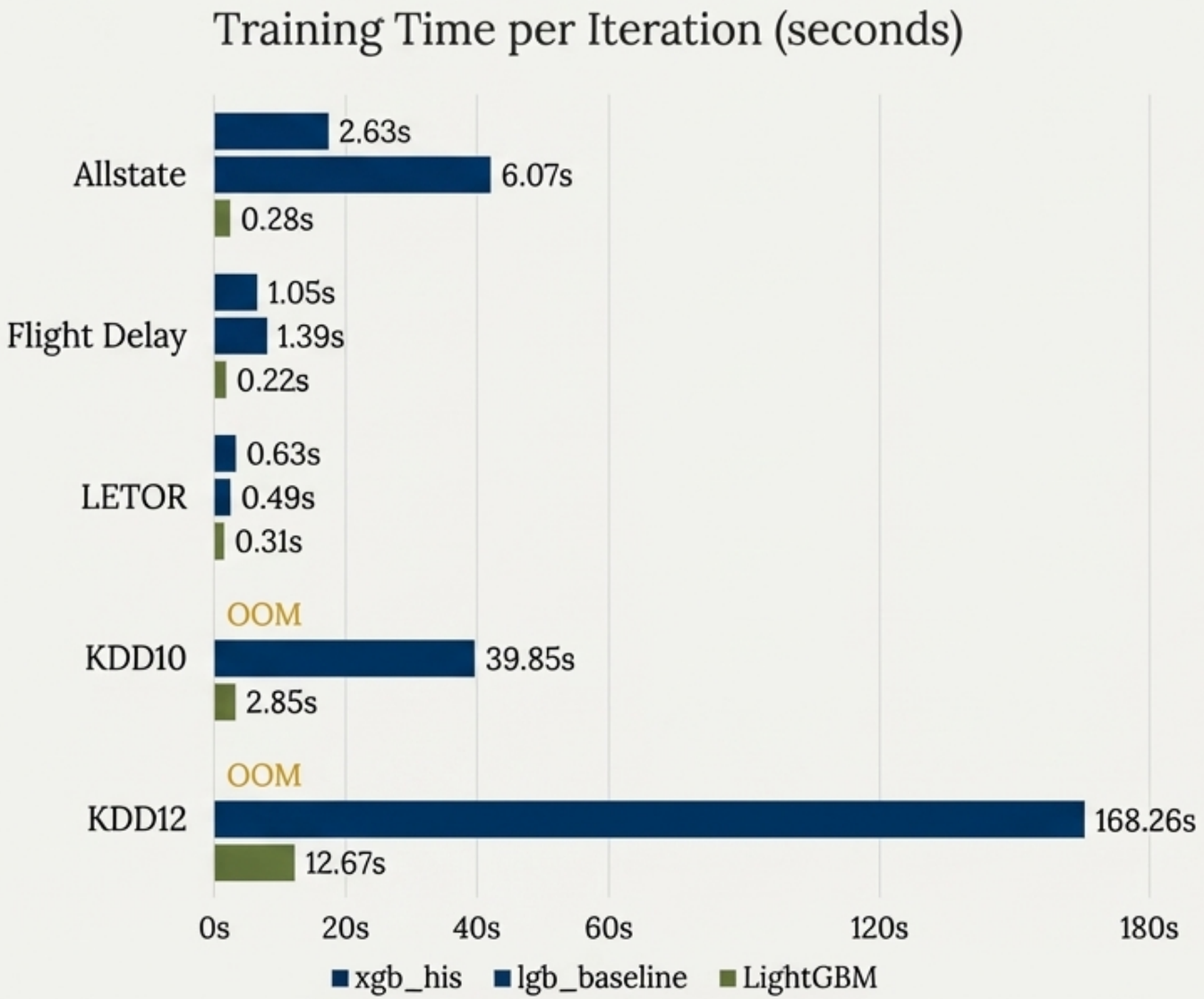


24-core server

256GB memory

16 threads

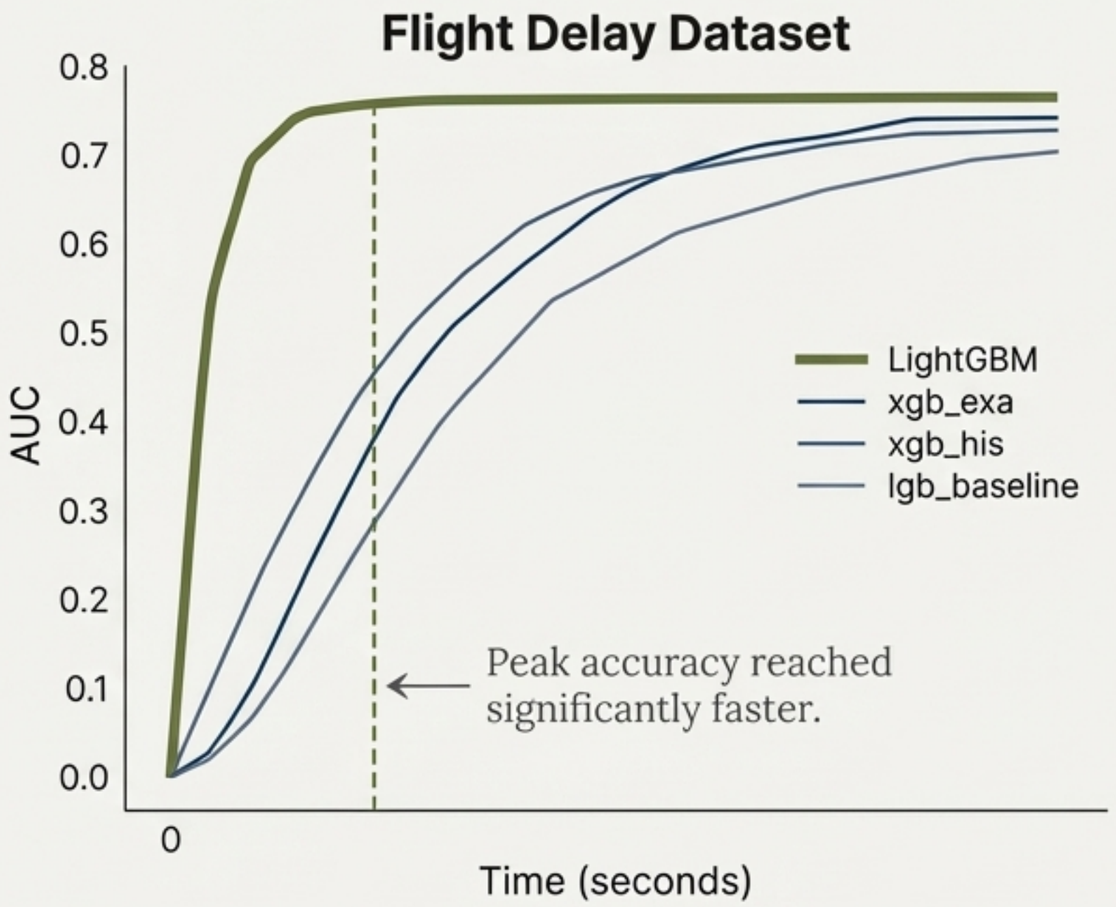
LightGBM achieves up to 21x speed-up with no loss in accuracy.

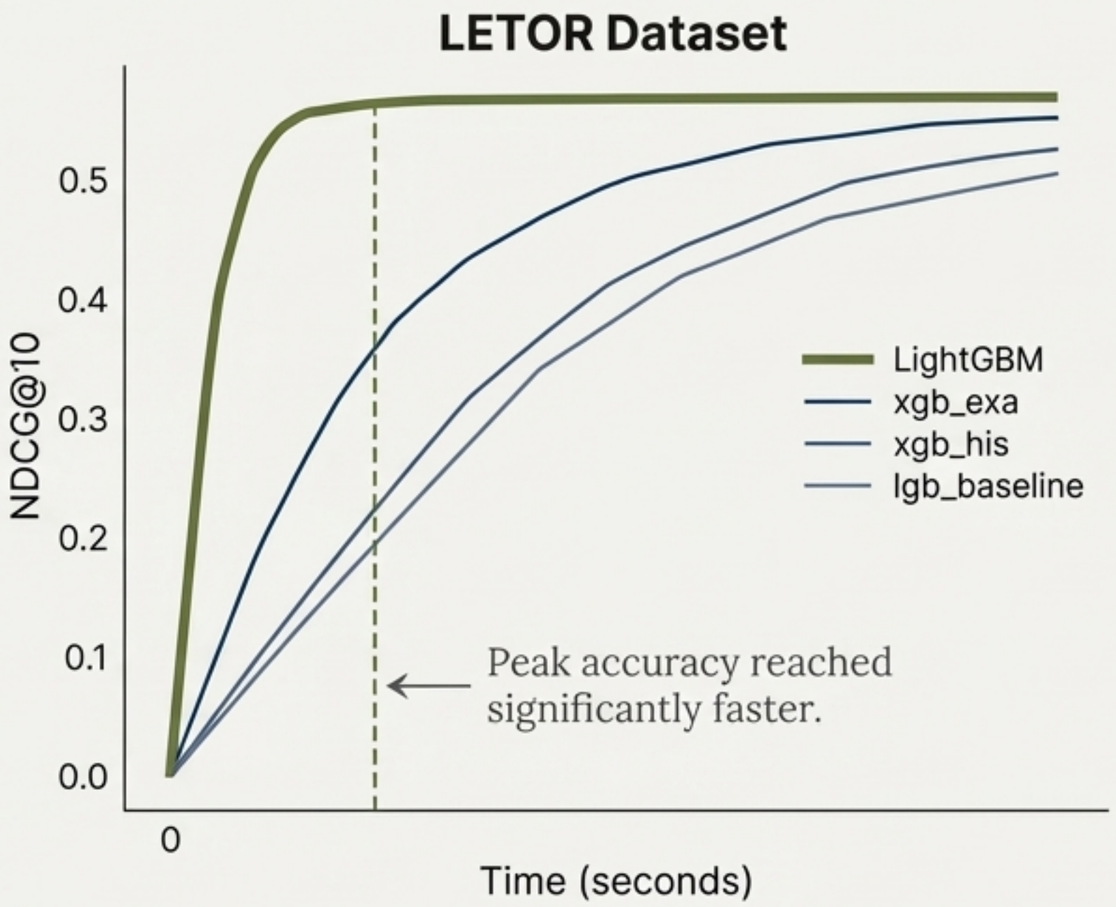


Key Takeaways

- **Speed-up vs lgb_baseline**: Allstate (**21x**), Flight Delay (**6x**), KDD10 (**14x**), KDD12 (**13x**).
- **Accuracy Maintained**: Test accuracy (AUC/NDCG) is almost identical to baselines across all datasets.
- **Memory Efficiency**: XGBoost (`xgb_his) runs out of memory (OOM) on the two largest datasets, while LightGBM handles them easily.

Reaching peak accuracy in a fraction of the time



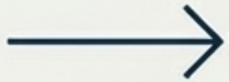


Both GOSS and EFB deliver significant, independent gains.

EFB reduces feature processing cost.

lgb_baseline: 6.07s

EFB_only: 0.71s



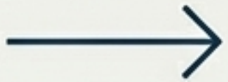
~8.5x speed-up

On KDD10, EFB reduces time from 39.85s to 6.33s (~6.3x speed-up).

GOSS reduces data instance cost.

EFB_only: 0.71s

LightGBM: 0.28s



~2.5x further speed-up

GOSS provides a consistent ~2x speed-up across datasets by using only 10-20% of the data.

GOSS is a more effective sampling method, consistently outperforming standard Stochastic Gradient Boosting (SGB) in accuracy at the same sampling ratio.

The new standard for large-scale gradient boosting.

The Challenge

Traditional GBDTs are too slow and memory-intensive for large datasets due to the $O(\#data \times \#feature)$ complexity of histogram building.

The Innovations

LightGBM introduces two targeted solutions:

- **GOSS** to intelligently reduce the number of data instances.
- **EFB** to intelligently reduce the number of features.

The Result

A highly efficient GBDT that is up to **21x faster** than its predecessors, uses less memory, and achieves the same high level of accuracy.

LightGBM: Faster training, same accuracy. Ready to use.

By addressing the core computational bottlenecks of GBDT, LightGBM enables data scientists and engineers to train highly accurate models on massive datasets more efficiently than ever before.

It represents a significant step forward in the evolution of gradient boosting frameworks.

The complete open-source code is available on GitHub.
`github.com/Microsoft/LightGBM`