LightGBM

Ensemble Learning:
Light GBM

Pilsung Kang
School of Industrial Management Engineering
Korea University

Light GBM

Ke et al. (2017)

- Motivation
  - ✓ Conventional GBM need to, for every feature, scan all the data instances to estimate the information gain of all the possible split points
- Idea
  - ✓ To reduce the number of data instances and the number of features
    - Gradient-based One-Side Sampling (GOSS)
      - Data instances with different gradients play different roles in the computation of information gain
      - Keep instances with large gradients and randomly drop instances with small gradients
    - Exclusive Feature Bundling (EFB)
      - In a sparse feature space, many features are (almost) exclusive, i.e., they rarely take nonzero values simultaneously (ex: one-hot encoding)
      - Bundling these exclusive features does not degenerate the performance

Light GBM

- Gradient-based One-sided Sampling (GOSS)

XGBoost

Algorithm 1: Histogram-based Algorithm

Input: I: training data, d: max depth
Input: m: feature dimension
$nodeSet \leftarrow \{0\}$ ▷ tree nodes in current level
$rowSet \leftarrow \{\{0, 1, 2, ...\}\}$ ▷ data indices in tree nodes
for $i = 1$ ***to*** d **do**
  for $node$ ***in*** $nodeSet$ **do**
    usedRows $\leftarrow rowSet[node]$
    for $k = 1$ ***to*** m **do**
      $H \leftarrow$ new Histogram()
      ▷ Build histogram
      for j ***in*** $usedRows$ **do**
        bin $\leftarrow I$.f[k][j].bin
        H[bin].y $\leftarrow H$[bin].y + I.y[j]
        H[bin].n $\leftarrow H$[bin].n + 1
      Find the best split on histogram H.
      ...
  Update $rowSet$ and $nodeSet$ according to the best split points.
  ...

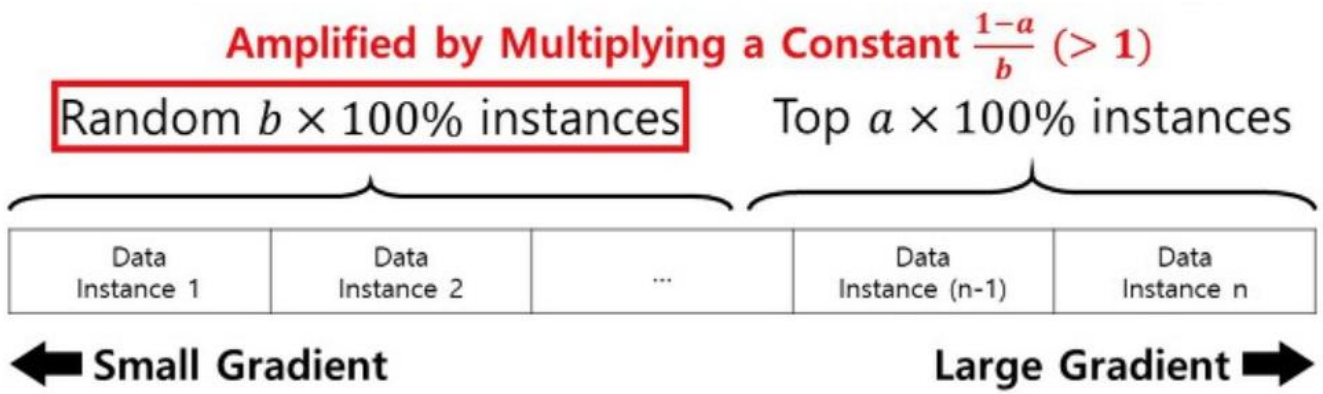
LightGBM

Algorithm 2: Gradient-based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models $\leftarrow$ { }, fact $\leftarrow \frac{1-a}{b}$
topN $\leftarrow$ a $\times$ len(I) , randN $\leftarrow$ b $\times$ len(I)
for $i = 1$ ***to*** d **do**
  preds $\leftarrow$ models.predict(I)
  g $\leftarrow loss(I$, preds), w $\leftarrow$ {1,1,...}
  sorted $\leftarrow$ GetSortedIndices(abs(g))
  topSet $\leftarrow$ sorted[1:topN]
  randSet $\leftarrow$ RandomPick(sorted[topN:len(I)], randN)
  usedSet $\leftarrow$ topSet + randSet
  w[randSet] $\times =$ fact ▷ Assign weight $fact$ to the small gradient data.
  newModel $\leftarrow$ L(I[usedSet], $-$ g[usedSet], w[usedSet])
  models.append(newModel)

Light GBM

- Gradient-based One-sided Sampling (GOSS)



https://cdm98.tistory.com/m/31

Light GBM

- Exclusive Feature Bundling (EFB)

Algorithm 3: Greedy Bundling

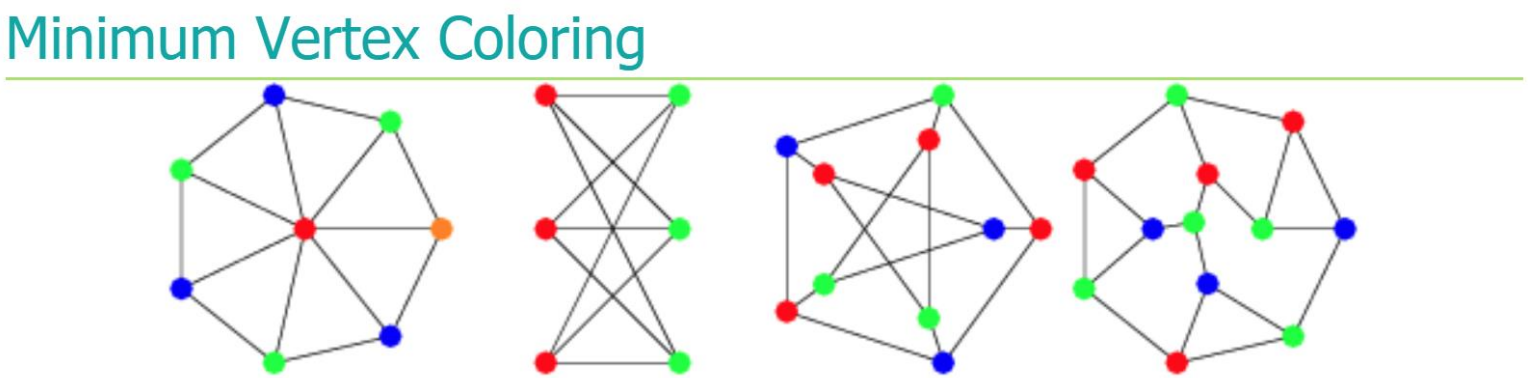
Input: F: features, K: max conflict count
Construct graph G
searchOrder $\leftarrow$ G.sortByDegree()
bundles $\leftarrow$ {}, bundlesConflict $\leftarrow$ {}
for i ***in*** *searchOrder* **do**
    needNew $\leftarrow$ True
    for $j = 1$ ***to*** *len(bundles)* **do**
        cnt $\leftarrow$ ConflictCnt(bundles[j],F[i])
        if *cnt* + *bundlesConflict[i]* $\leq K$ **then**
            bundles[j].add(F[i]), needNew $\leftarrow$ False
            break
    if *needNew* **then**
        Add $F[i]$ as a new bundle to *bundles*
Output: *bundles*

Algorithm 4: Merge Exclusive Features

Input: $numData$: number of data
Input: F: One bundle of exclusive features
binRanges $\leftarrow$ {0}, totalBin $\leftarrow$ 0
for f ***in*** F **do**
    totalBin += f.numBin
    binRanges.append(totalBin)
newBin $\leftarrow$ new Bin(numData)
for $i = 1$ ***to*** $numData$ **do**
    newBin[i] $\leftarrow$ 0
    for $j = 1$ ***to*** *len(F)* **do**
        if *F[j].bin[i]* $\neq 0$ **then**
            newBin[i] $\leftarrow$ F[j].bin[i] + binRanges[j]
Output: $newBin, binRanges$

Light GBM

- Exclusive Feature Bundling (EFB)
  - ✓ Can be formulated as a Graph coloring problem
    - Construct a Graph (V, E)
      - V: feature
      - E: total conflicts between features



Light GBM

- Exclusive Feature Bundling (EFB)
  - ✓ Greedy bundling example

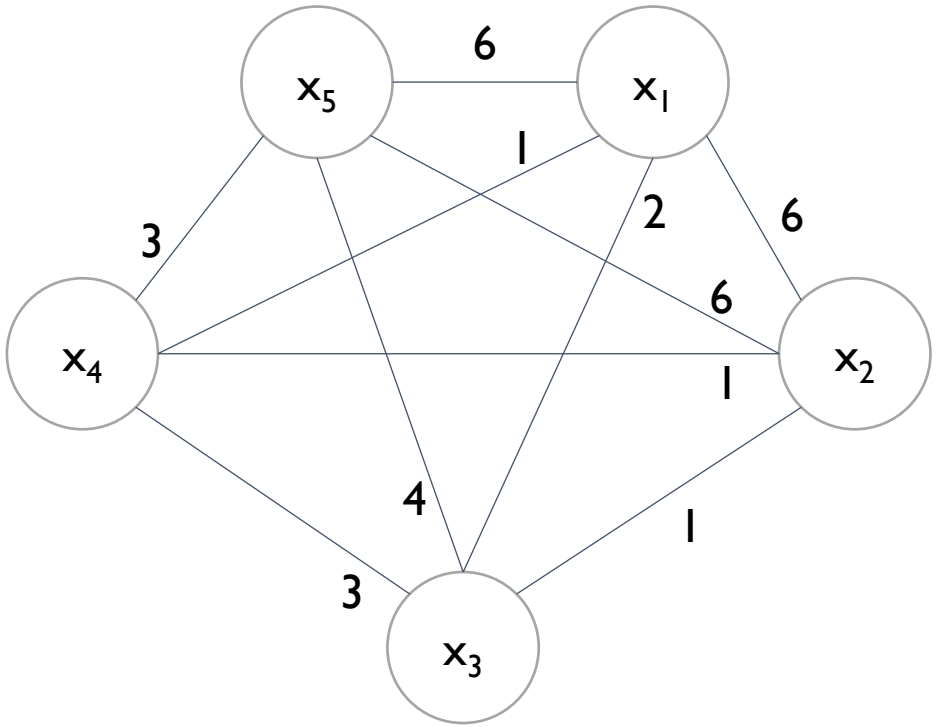
	x_1	x_2	x_3	x_4	x_5
l_1	1	1	0	0	1
l_2	0	0	1	1	1
l_3	1	2	0	0	2
l_4	0	0	2	3	1
l_5	2	1	0	0	3
l_6	3	3	0	0	1
l_7	0	0	3	0	2
l_8	1	2	3	4	3
l_9	1	0	1	0	0
l_{10}	2	3	0	0	2

	x_1	x_2	x_3	x_4	x_5
x_1	-	6	2	1	6
x_2	6	-	1	1	6
x_3	2	1	-	3	4
x_4	1	1	3	-	3
x_5	6	6	4	3	-

	x_5	x_1	x_2	x_3	x_4
d	19	15	14	10	8

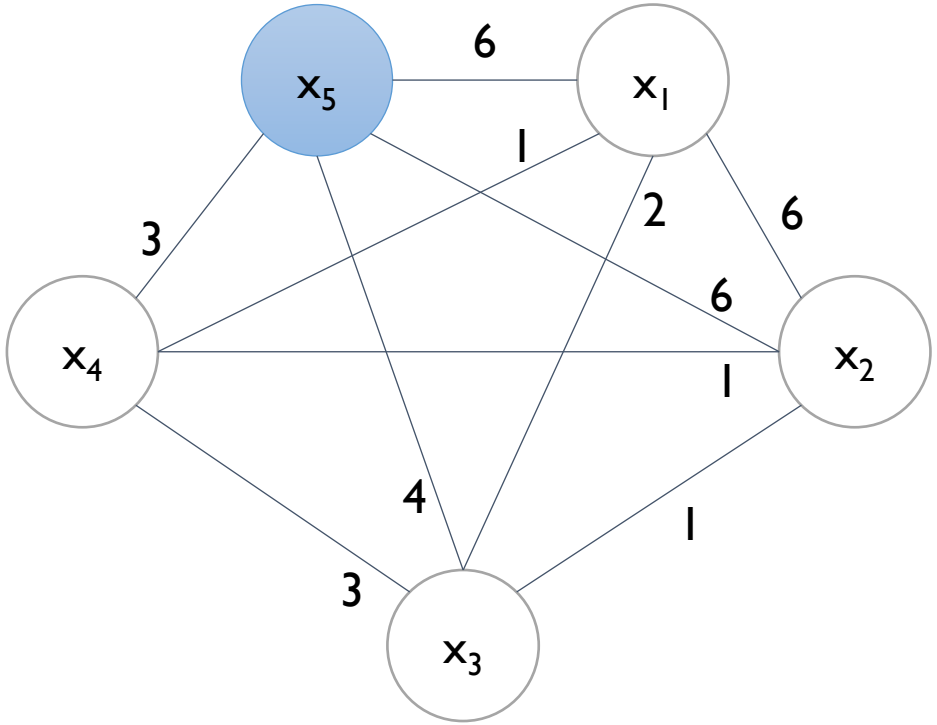
Light GBM

- Exclusive Feature Bundling (EFB)
  - ✓ Greedy bundling example (cut-off = 0.2)



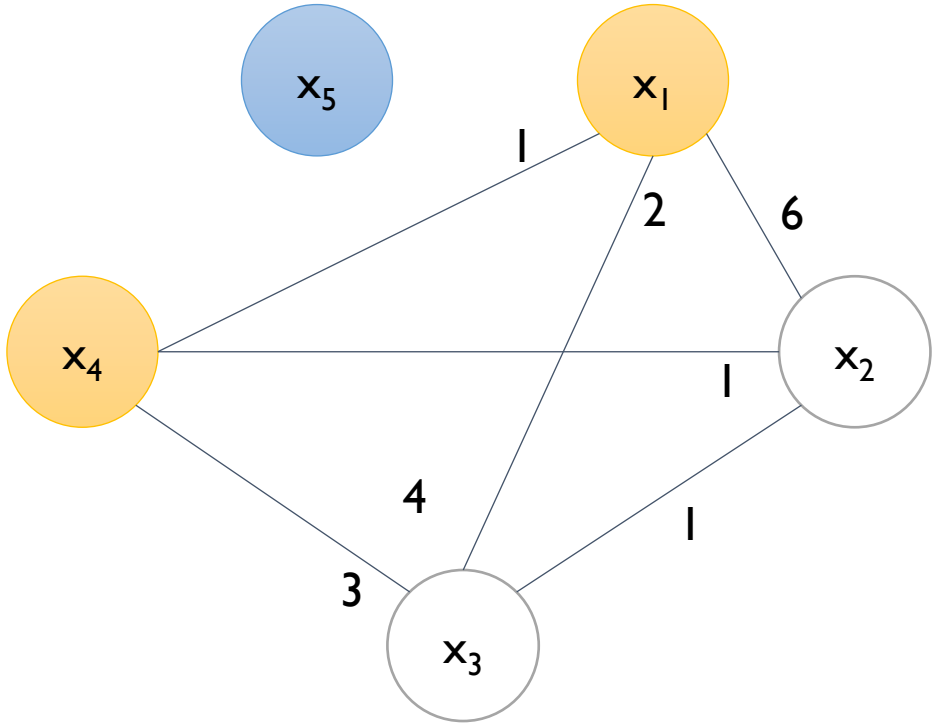
Light GBM

- Exclusive Feature Bundling (EFB)
  - ✓ Greedy bundling example (cut-off = 0.2)



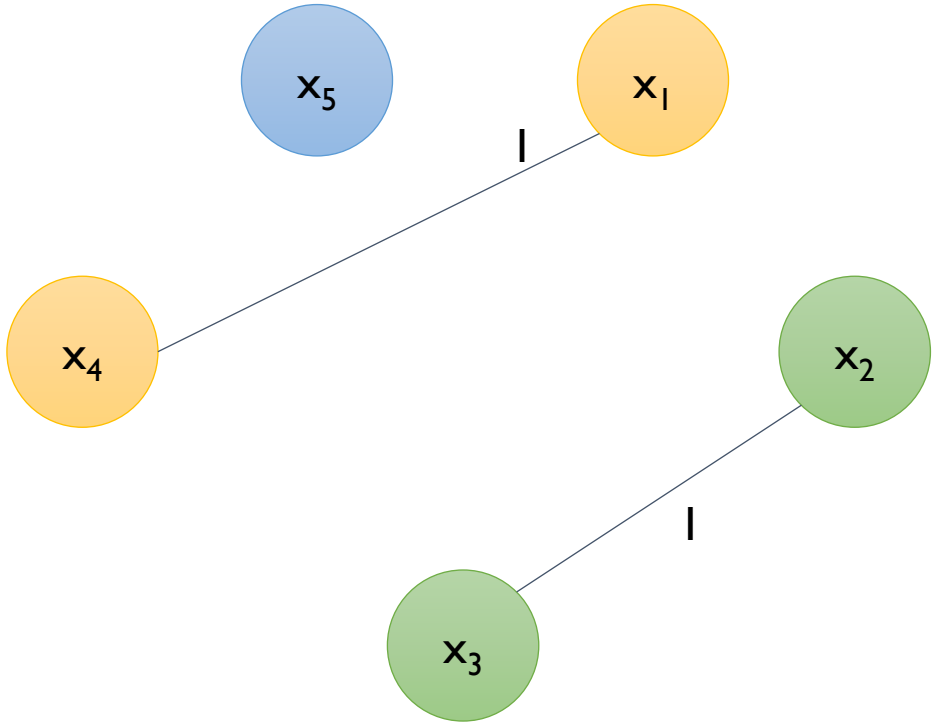
Light GBM

- Exclusive Feature Bundling (EFB)
  - ✓ Greedy bundling example (cut-off = 0.2)



Light GBM

- Exclusive Feature Bundling (EFB)
  - ✓ Greedy bundling example (cut-off = 0.2)



Light GBM

- Exclusive Feature Bundling (EFB)
  - ✓ Greedy bundling example (cut-off = 0.2)

	x_1	x_2	x_3	x_4	x_5
I_1	1	1	0	0	1
I_2	0	0	1	1	1
I_3	1	2	0	0	2
I_4	0	0	2	3	1
I_5	2	1	0	0	3
I_6	3	3	0	0	1
I_7	0	0	3	0	2
I_8	1	2	3	4	3
I_9	1	0	1	0	0
I_{10}	2	3	0	0	2

	x_5	x_1	x_4	x_2	x_3
I_1	1	1	0	1	0
I_2	1	0	1	0	1
I_3	2	1	0	2	0
I_4	1	0	3	0	2
I_5	3	2	0	1	0
I_6	1	3	0	3	0
I_7	2	0	0	0	3
I_8	3	1	4	2	3
I_9	0	1	0	0	1
I_{10}	2	2	0	3	0

Light GBM

- Exclusive Feature Bundling (EFB)
  - ✓ Exclusive feature merging
    - Add offsets to the original values of the features

	x_5	x_1	x_4	x_2	x_3
I_1	1	1	0	1	0
I_2	1	0	1	0	1
I_3	2	1	0	2	0
I_4	1	0	3	0	2
I_5	3	2	0	1	0
I_6	1	3	0	3	0
I_7	2	0	0	0	3
I_8	3	1	4	2	3
I_9	0	1	0	0	1
I_{10}	2	2	0	3	0

	x_5	x_{14}	x_{23}
I_1	1	1	1
I_2	1	4	4
I_3	2	1	2
I_4	1	6	5
I_5	3	2	1
I_6	1	3	3
I_7	2	0	6
I_8	3	1	2
I_9	0	1	4
I_{10}	2	2	3

Add the offset 3 to the nonzero values of x_4

Add the offset 3 to the nonzero values of x_3

Conflict:
Use the value of x_1

Conflict:
Use the value of x_2

Light GBM

- Experiments
  - ✓ Dataset description

Table 1: Datasets used in the experiments.

Name	$\#data$	$\#feature$	Description	Task	Metric
Allstate	12 M	4228	Sparse	Binary classification	AUC
Flight Delay	10 M	700	Sparse	Binary classification	AUC
LETOR	2M	136	Dense	Ranking	NDCG [4]
KDD10	19M	29M	Sparse	Binary classification	AUC
KDD12	119M	54M	Sparse	Binary classification	AUC

  - ✓ Training time

	xgb_exa	xgb_his	lgb_baseline	EFB_only	**LightGBM**
Allstate	10.85	2.63	6.07	0.71	**0.28**
Flight Delay	5.94	1.05	1.39	0.27	**0.22**
LETOR	5.55	0.63	0.49	0.46	**0.31**
KDD10	108.27	OOM	39.85	6.33	**2.85**
KDD12	191.99	OOM	168.26	20.23	**12.67**

Light GBM

- Experiments
  - ✓ Overall accuracy

	xgb_exa	xgb_his	lgb_baseline	SGB	**LightGBM**
Allstate	0.6070	0.6089	0.6093	0.6064 ± 7e-4	**0.6093 ± 9e-5**
Flight Delay	0.7601	0.7840	0.7847	0.7780 ± 8e-4	**0.7846 ± 4e-5**
LETOR	0.4977	0.4982	0.5277	0.5239 ± 6e-4	**0.5275 ± 5e-4**
KDD10	0.7796	OOM	0.78735	0.7759 ± 3e-4	**0.78732 ± 1e-4**
KDD12	0.7029	OOM	0.7049	0.6989 ± 8e-4	**0.7051 ± 5e-5**

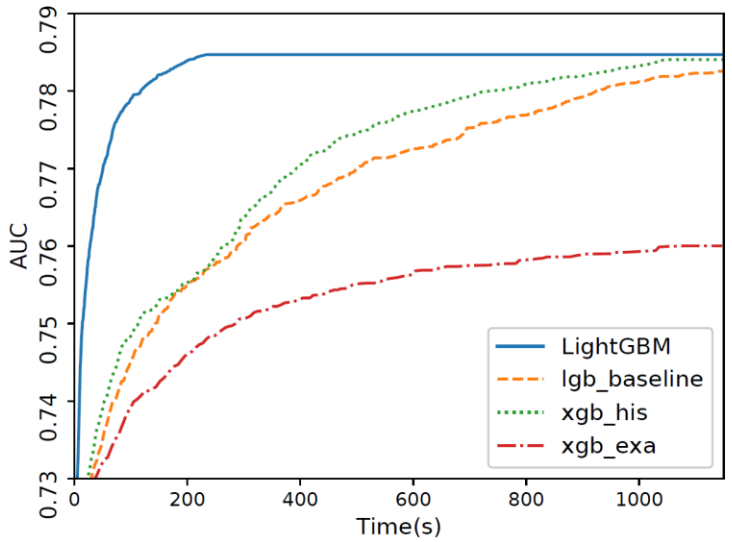


Figure 1: Time-AUC curve on Flight Delay.

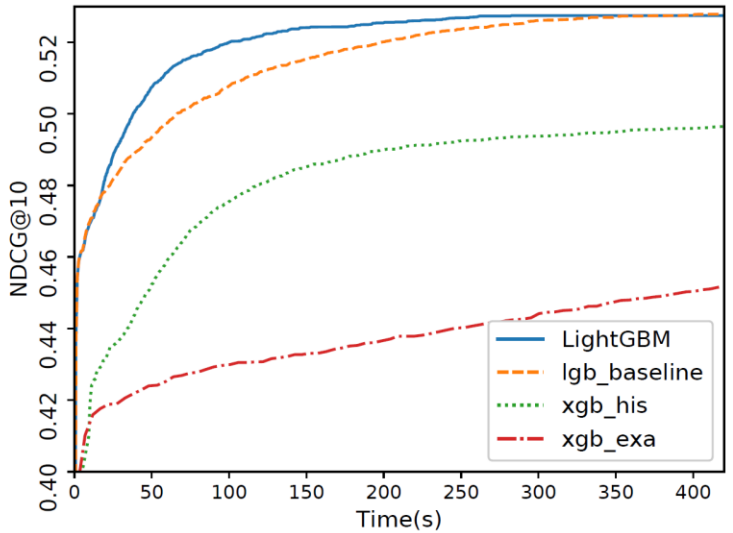


Figure 2: Time-NDCG curve on LETOR.


Q & A