

LightGBM

LightGBM

<https://www.notion.so/LightGBM-4b71bdf8a4d1459e81113c20a59909c3?pvs=4>

Background

Light Gradient Boosting Model

Light

'가벼운' 모형 → 매우 빠른 속도 + 적은 메모리



Boosting

to improve 라는 뜻

Weak learner를 여러 번 사용하여 성능을 높이는 방법

분석 절차

1. Weak learner를 이용해 분류한 후, n 개의 학습 데이터 중

- 분류가 올바르게 된 학습 데이터 → 가중치 감소
- 오분류 된 학습 데이터 → 가중치 증가
⇒ 오분류된 데이터가 추출될 확률 증가!

2. 위를 M 번 반복

이 때, L_i 를 i 번째 learner라고 하면, $L = \Sigma L_i$

⇒ **addictive model** : 비모수 회귀 (함수의 형태를 가정하지 않는 회귀 모형)

▼ Bagging, Pasting etc.

Gradient Boosting

1. 모든 features
2. 모든 data instances

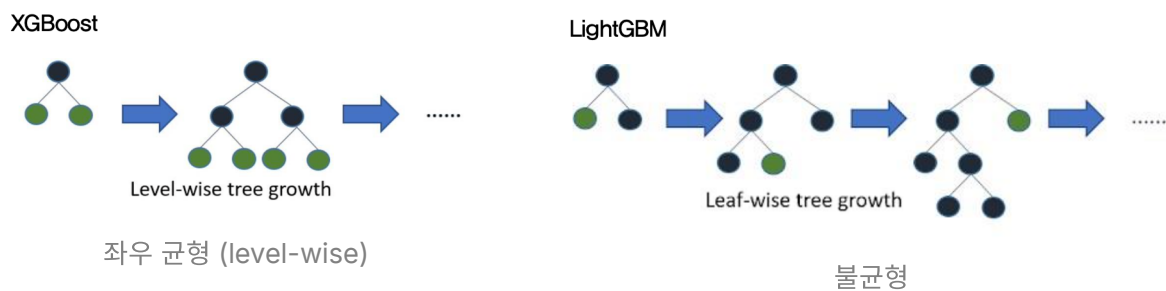
→ information gain을 추정하는 방식으로 가능한 split point를 탐색 (완탐...?)

LGBM 특성

(XGBoost의 변형)

1. Leaf-wise (Best-first) Tree (imbalanced)

- 동일한 개수의 leaf node로 비교적 **효과적** 이고 **높은 수준의 성능**
- 손실 함수를 가장 크게 줄일 수 있는 부분에서 나누기 (성능이 좋은 쪽으로 수준 (level) 높이기)



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


- 모든 표본을 사용하지 않고도 model의 성능 유지 가능
- 각 표본에 대한 미분 값(**gradient**)의 절댓값을 기준
 - 일정 크기 이상인 경우: 모든 표본 사용
 - 나머지: 일부분만 사용

3. Exclusive Feature Bundling (EFB)

- 2개의 변수가 동시에 0이 아닌 값을 가질 확률이 매우 낮을 때 사용
 - x_1 : [0, 10]
 - x_2 : [0, 20]
 - x_3 : [0, 30] ($x_1 + x_2$)

[Ensemble] Light GBM, 마이크로소프트의 부스팅

Light GBM, 마이크로소프트의 부스팅 Ensemble GBM은 최적의 Information Gain을 갖기위해서 모든 split point를 살펴봐야했습니다. XGBoost에서는 전체데이터를 버킷단위

 <https://exupery-1.tistory.com/184>

1. Greedy Bundling

- Graph coloring problem
- **exclusive**: 동시에 0을 가지지 않음

1. weighted edges를 기반으로 Graph 구성 (weight는 feature사이에 conflict에 대한 값)

여기서 **conflict**가 뭔지 정확하게 모르겠어요,,, ; 두 변수 모두 0이 아닐 때

2. 그래프의 데이터를 weight 기준 내림차순으로 정렬

3. 각 feature를 확인하고 maximal conflict rate를 기준으로 bundling

2. Merge Exclusive Features

- 기준 변수 + offset
 - conflict 된 부분은 **기준 변수**로
 - 아닌 부분은 **offset**으로

	x1	x2	x12
0	1	0	1
1	3	0	3
2	4	0	4
3	0	18	28
4	0	13	23
5	0	20	30
6	1	0	1
7	2	0	2
8	0	5	15
9	10	0	10
10	7	9	7
11	9	0	9

기준변수를 x_1 으로 잡고 merge를 시킬 때

x_1 에 값이 있으면 x_1 그대로 가져옵니다.

x_2 에 값이 있으면 Offset을 더해서 넣습니다.

여기서 offset은 10이 됩니다. (x_1 는 10까지있으므로)

만약 충돌된 값이 있으면 기준변수 x_1 를 그대로 가져옵니다. (10번째)

Algorithms4에서는 이러한 방식으로 Merge를 하게됩니다.

여기서 충돌된 10번째 데이터 빼고는 손실되는 데이터가 없습니다.

4. Optimal Split for Categorical Features

- to split on a categorical feature by partitioning its categories into 2 subsets
 - k 개의 범주가 있다면, $2^{k-1} - 1$ 개의 파티션 생성 가능

- The basic idea is to sort the categories according to the **training objective** at each split. More specifically, LightGBM sorts **the histogram** (for a categorical feature) according to its accumulated values (`sum_gradient / sum_hessian`) and then finds the best split on the **sorted histogram**.



계산 속도가 훨씬 빠르고, 성능 우수

단점

- 데이터의 크기가 작을 때는 과대적합(`overfitting`) 발생
 - `10,000` 개 이상의 observation(row)가 있을 때 사용 권장
- Overfitting에 민감, `하이퍼파라미터` 튜닝 시 주의
 - 하이퍼파라미터에 따른 성능 차이가 좀 있는 편



LightGBM 하이퍼파라미터 튜닝

`num_leaves` 의 개수를 중심으로 `min_child_samples` (min_data_in_leaf), `max_depth` 를 함께 조절하면서 모델의 복잡도를 줄이는 것이 기본 튜닝 방안

- `num_leaves` 를 늘리면 정확도가 높아지지만 트리가 깊어지고 과적합되기 쉬움
- `min_child_samples` (min_data_in_leaf)를 크게 설정하면 트리가 깊어지는 것을 방지
- `max_depth` 는 명시적으로 깊이를 제한, 과대 적합 방지

! `learning_rate`을 줄이면서 `n_estimator`를 크게 (Boosting 기본적인 튜닝 방안)