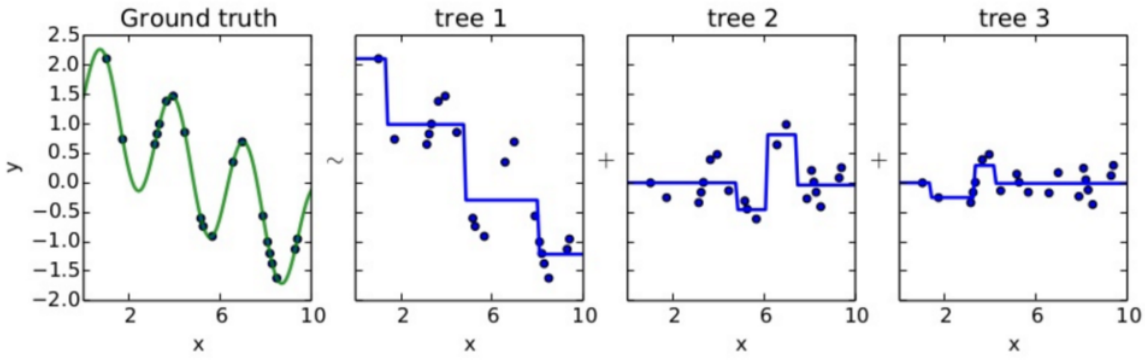
XGBoost

1. GBM?

xgboost는 트리기반 모델로 앙상블 기법 중 부스팅 또 부스팅 중 Gradient boost(GBM)를 사용한다.



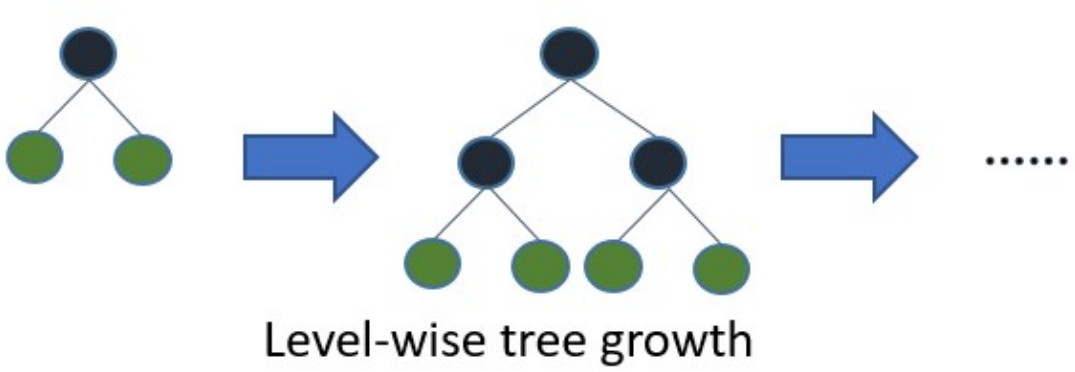
- GBM이란 여러 약분류기를 순차적으로 학습하며 각 step(iteration)에서 예측된 데이터에 대하여 가중치를 부여를 통해 오류를 개선해 나간다.
- GBM의 경우 가중치 업데이트에 경사하강법을 이용하는 것이 큰 특징이다. 즉, 오류값을 최소화하는 방향으로 가중치 값을 업데이트 한다.
- 또한 CART(Classification And Regression Trees)기반의 알고리즘으로 분류 뿐 아니라 회귀도 가능하다.
- 하지만 알고리즘의 특성상 긴 학습시간과 과적합이 단점이다.

2. 그렇다면 XGBoost는?

1. **병렬 처리 지원**

   먼저 병렬학습을 통해 기존 BGM보다 빠르게 학습이 가능해 졌다.

2. **가지치기(Tree pruning)**



XGB는 지정된 최대 깊이 까지 level_wise 방식으로 분할 한 다음 이득이 없는 가지를 쳐내는 방식을 사용한다.

3. **regularization**

$$\text{XGBoost loss function} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

GBM은 잔차(residual)를 계속 줄여나가는 알고리즘이기 때문에 과적합에 취약하다는 문제가 있다.

XGB에서는 이런 문제를 해결하기위해 regularization term을 추가했다.

4. **Shrinkage and Column Subsampling**
   - Shrinkage

     부스팅 트리의 각 단계 이후 마다 새로운 가중치를 요인으로 Scaling
   - Column Subsampling

     Random Forest에서 차용

     각 단계에서 컬럼을 sub-sampling하여 학습
5. **결측치 자체 처리**
6. **custom opimization 가능**

3. Split finding 알고리즘

1. **exact greedy algorithm**

   모든 feature에 대해 모든 가능한 split을 탐색하는 방법(데이터 sorting 필요)
2. **approximate algorithm**

   메모리에 데이터가 적합되지 못할 때 사용

Feature 분포의 percentiles에 따라 후보분할지점 제시

연속적인 Feature를 후보지점으로 나뉜 Bucket 에 매핑하고 통계량 집계

통계량을 바탕으로 제안된 방안중 최적의 솔루션을 찾음

3. **Weighted Quantile Sketch**

   weight가 반영된 split 방법

   일반적으로 quantile을 계산할 때, 각 구간별로 관측치의 수를 동일하게 하지만 XGB는 모든 관측치에 Weight를 주고, 그 weight들의 합을 구간별로 동일하게 유지하는 방식으로 진행

4. **Sparsity_aware split finding**

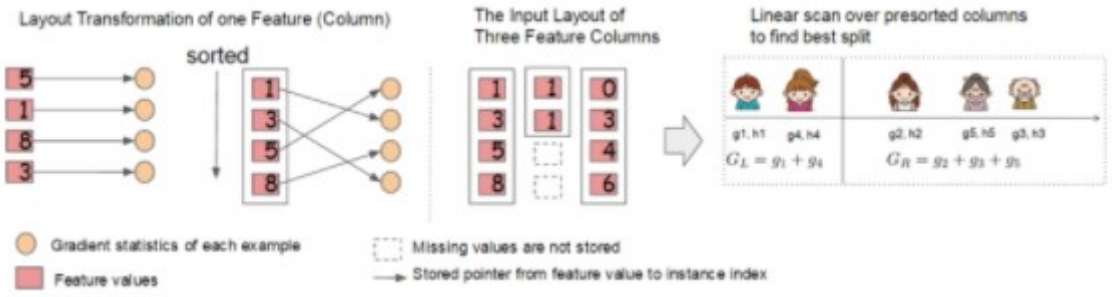
   결측치가 있는 데이터셋도 학습이 가능하게 함

   결측치를 전부 오른쪽, 왼쪽으로 보낸 뒤 더 좋은 split point를 찾아냄

   비교적 단순한 결측치 처리로 기존 알고리즘 보다 속도 향상

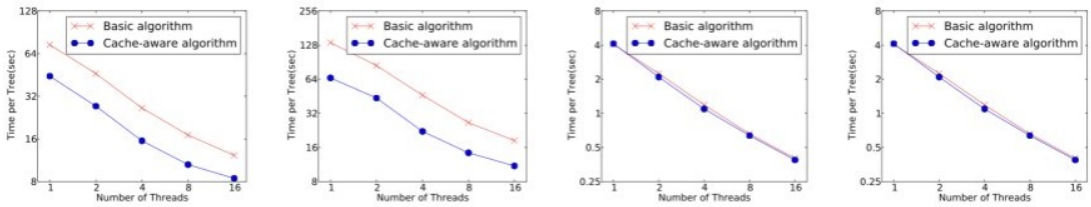
4. System 디자인

1. **column block for parallel learning**



정렬비용을 감소시키고 병렬적인 알고리즘을 가능하게함

2. **cache aware access**



(a) Allstate 10M (b) Higgs 10M (c) Allstate 1M (d) Higgs 1M

Figure 7: Impact of cache-aware prefetching in exact greedy algorithm. We find that the cache-miss effect impacts the performance on the large datasets (10 million instances). Using cache aware prefetching improves the performance by factor of two when the dataset is large.

CPU 캐시 인식을 사용하여 데이터의 크기가 커져도 빠른속도로 학습가능