Ensemble Learning

6주차
담당: 14기 박상준

1. Ensemble Methods

2. Ensemble Models
   - Random Forest
   - Adaboost
   - GBM
   - XgBoost

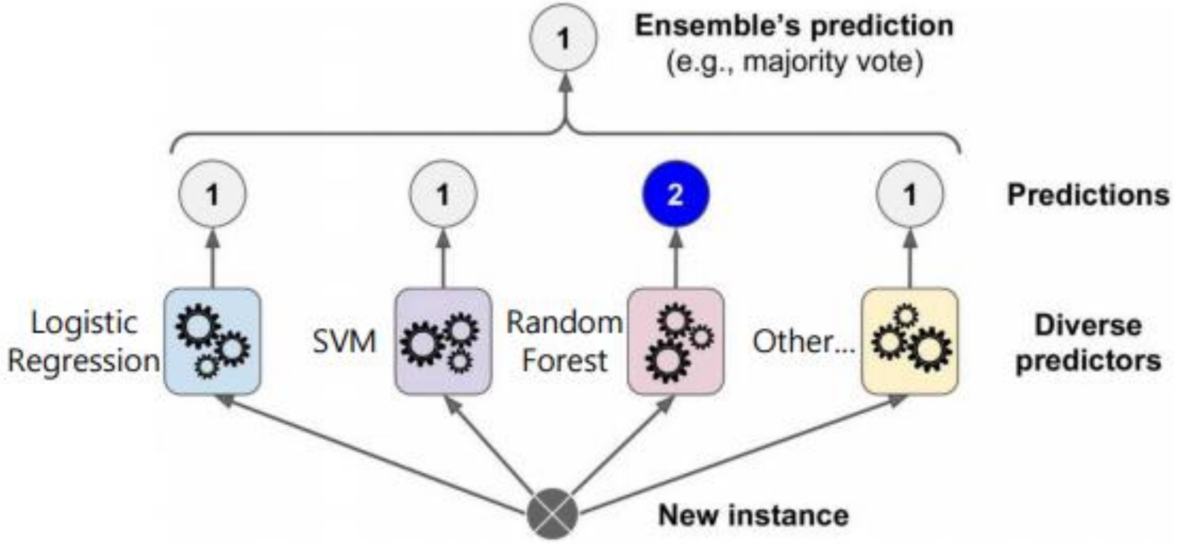
3. Coding Session

Ensemble Methods

‹ Voting › Hard Voting Soft Voting Weighted Voting	‹ Stacking › Meta level learning
‹ Bagging ›	‹ Boosting ›

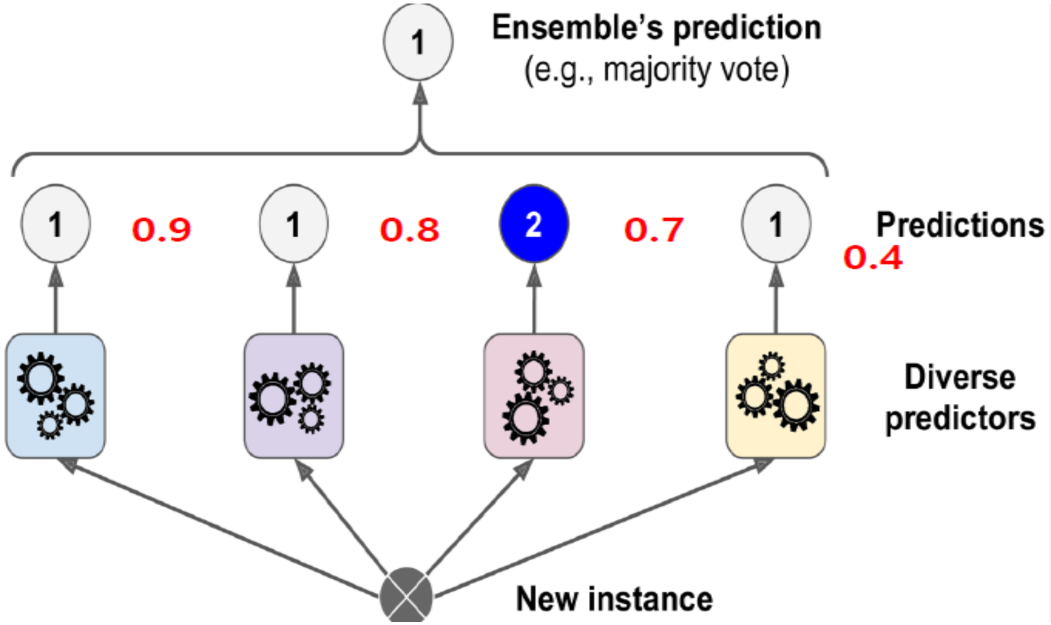
Ensemble Methods

Hard Voting



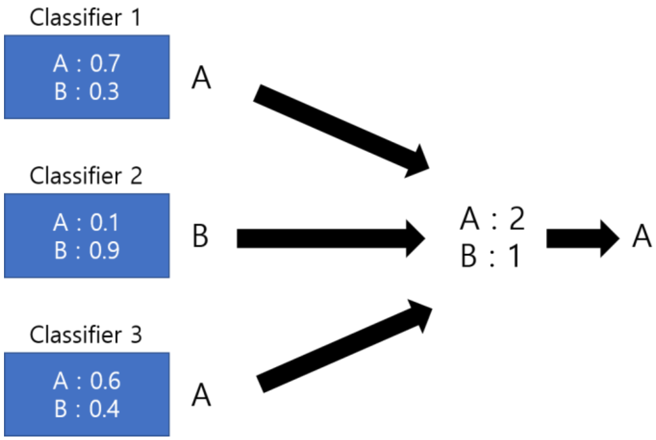
Ensemble Methods

Soft Voting

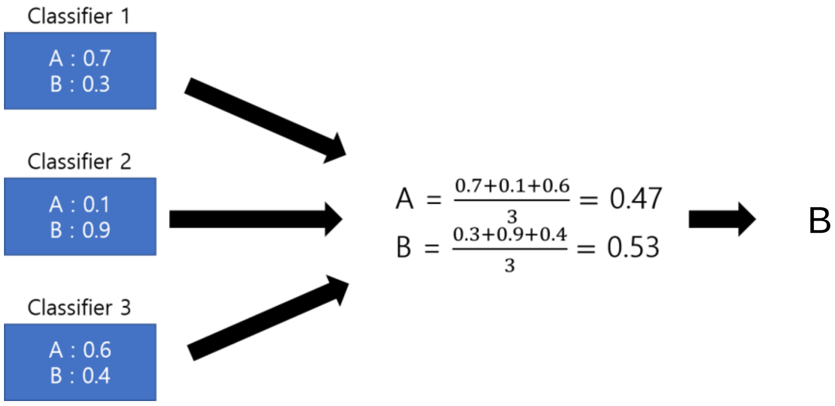


Ensemble Methods

Hard Voting

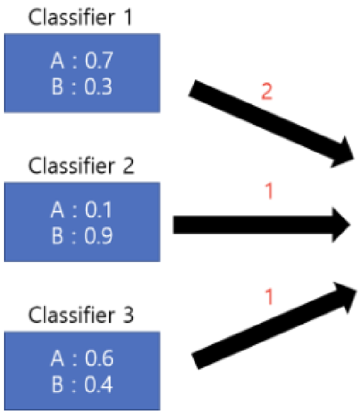


Soft Voting



Ensemble Methods

Weighted Voting



$$A = \frac{0.7\times2+0.1\times1+0.6\times1}{4} = 0.53$$

$$B = \frac{0.3\times2+0.9\times1+0.4\times1}{3} = 0.47$$

Soft Voting + Weighted Voting

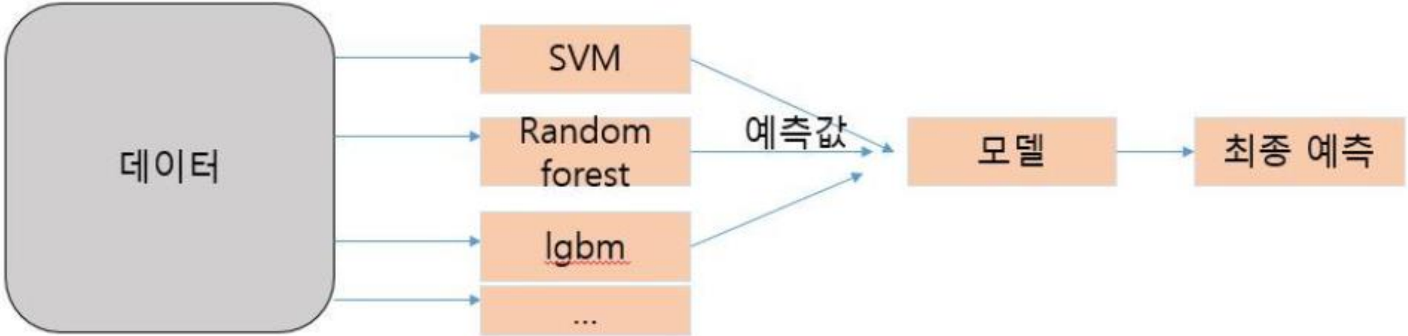
Ensemble Methods

‹ Voting › Hard Voting Soft Voting Weighted Voting	‹ Stacking › Meta level learning
‹ Bagging ›	‹ Boosting ›

Ensemble Methods

Stacking

개별 모델이 예측한 결과를 다시 input data로 사용



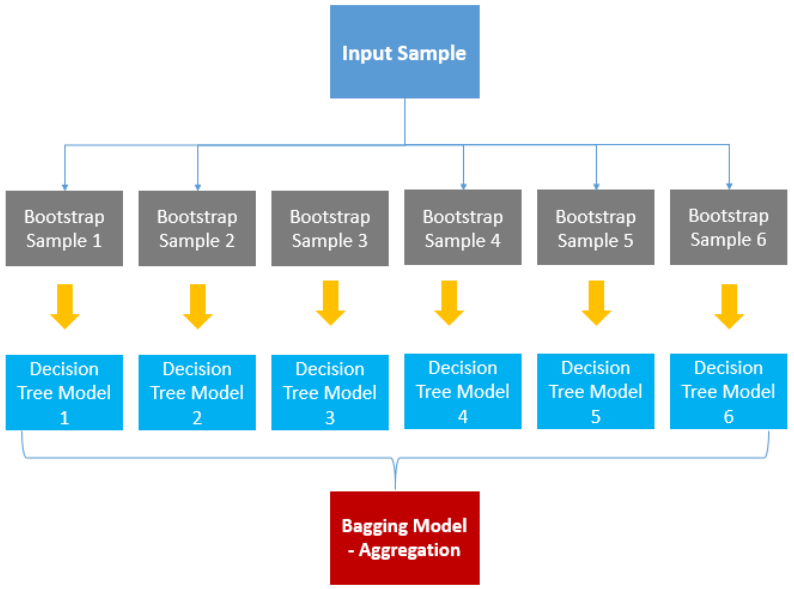
Ensemble Methods

‹ Voting › Hard Voting Soft Voting Weighted Voting	‹ Stacking › Meta level learning
‹ Bagging ›	‹ Boosting ›

Ensemble Methods

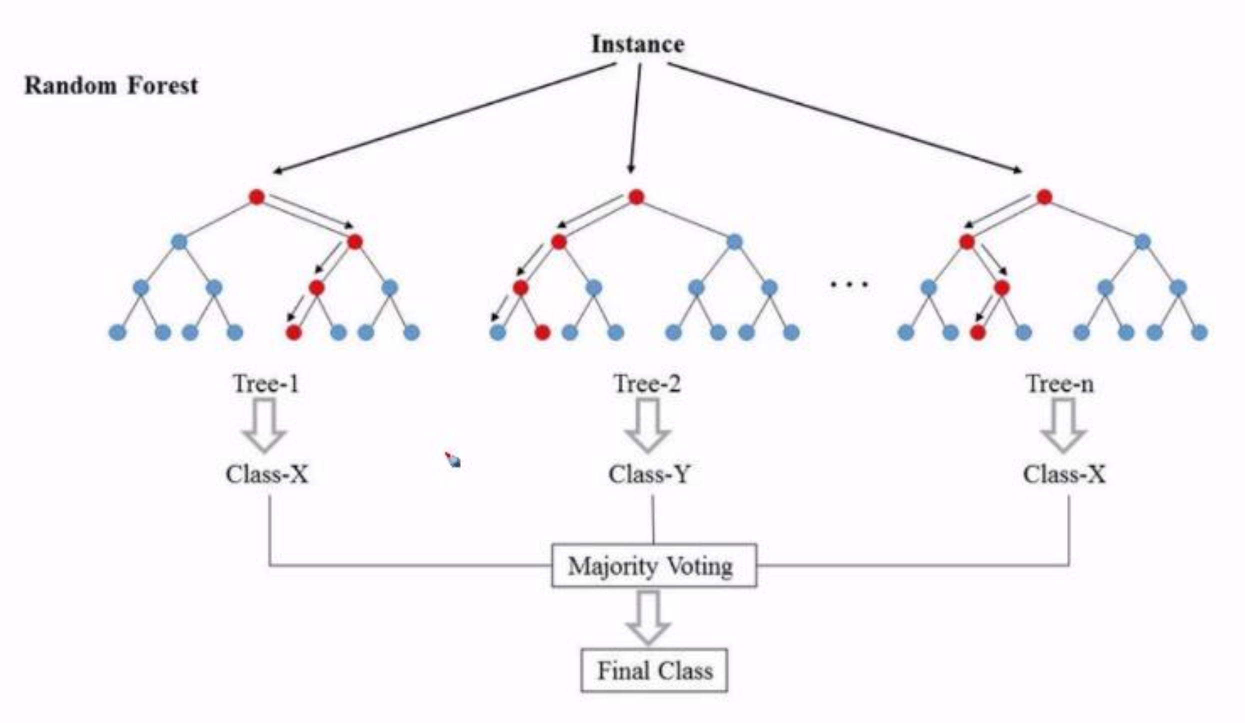
Bagging

Sampling with Replacement



Random Forest

Random Forest



Random Forest

장점

- 어디에 가져다 놓아도 중간 이상 가는 알고리즘
- 병렬 학습 방식으로 학습 속도가 빠른 편
- 입력 변수가 많아도 잘 작동함

단점

- 트리 개수, 깊이 같은 초모수를 잘못 설정할 경우 Overfitting의 가능성이 매우 높음

Coding Session

Random Forest

sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

- **n_estimators:** 복원 추출 횟수
- **Max_features(sqrt, log2, None):** 최적의 분할 기준 찾을 때 사용할 feature 개수
- **criterion(gini, entropy, log loss):** 트리 분할 기준 metric
- **max_depth:** Tree의 깊이(None이면 계속 갈라짐)
- **min_samples_split:** 노드 분할하기 위한 최소 sample 수
- **min_samples_leaf:** 리프 노드에 있는 최소 sample 수
- **max_leaf_nodes:** 리프 노드 최대 개수
- **max_samples:** 샘플링 비율

Ensemble Methods

‹ Voting ›	‹ Stacking ›
Hard Voting Soft Voting Weighted Voting	Meta level learning
‹ Bagging ›	**‹ Boosting ›**

Ensemble Methods

‹ Normal ›	‹ Bagging ›	‹ Boosting ›	
학생 A : 10개년 6,9,수능 문제 1회독	학생 B : 10개년 6,9,수능 전체 문제에서 80% 랜덤 복원 추출 -> 10번 반복	학생 C : 10개년 6,9,수능 전체 문제에서 80% 랜덤 복원 추출 이때, 틀린 문제는 반드시 포함해서 추출 -> 10번 반복	학생 D : 10개년 6,9,수능 전체 문제 1회독 -> 틀린 문제만 뽑아서 다시 1회독 -> 다시 틀린 문제만 뽑아서 1회독 -> 다시 틀린 문제만 뽑아서 1회독 . . . 안틀릴때까지 반복

Ensemble Methods

This Week

Adaptive Boosting (AdaBoost)

Gradient Boosting Machine (GBM)

Extreme Gradient Boosting Machine (XgBoost)

Next Week

Light Gradient Boosting Machine (LGBM)

Categorical Boosting Machine (CatBoost)

AdaBoost

Adaptive Boosting (Adaboost)

❖ AdaBoost

- 각 단계에서 새로운 base learner를 학습하여 이전 단계의 base learner의 단점을 보완
- Training error가 큰 관측치의 선택 확률(가중치)을 높이고, training error가 작은 관측치의 선택 확률을 낮춤
  - ✓ 오분류한 관측치에 보다 집중!
- 앞 단계에서 조정된 확률(가중치)을 기반으로 다음 단계에서 사용될 training dataset를 구성
- 다시 첫 단계로 감
- 최종 결과물은 각 모델의 성능지표를 가중치로 하여 결합 (앙상블)

AdaBoost

❖ AdaBoost algorithm

1. Set $W_i = \frac{1}{n}, i = 1,2,\dots,n$ (impose equal weight initially)

2. for $j = 1$ to m (m : number of classifiers)

Step 1: Find $h_j(x)$ that minimizes L_j (weighted loss function)

$$L_j = \frac{\sum_{i=1}^{n} w_i\, I(y_i \neq h_j(x))}{\sum_{i=1}^{n} w_i}$$

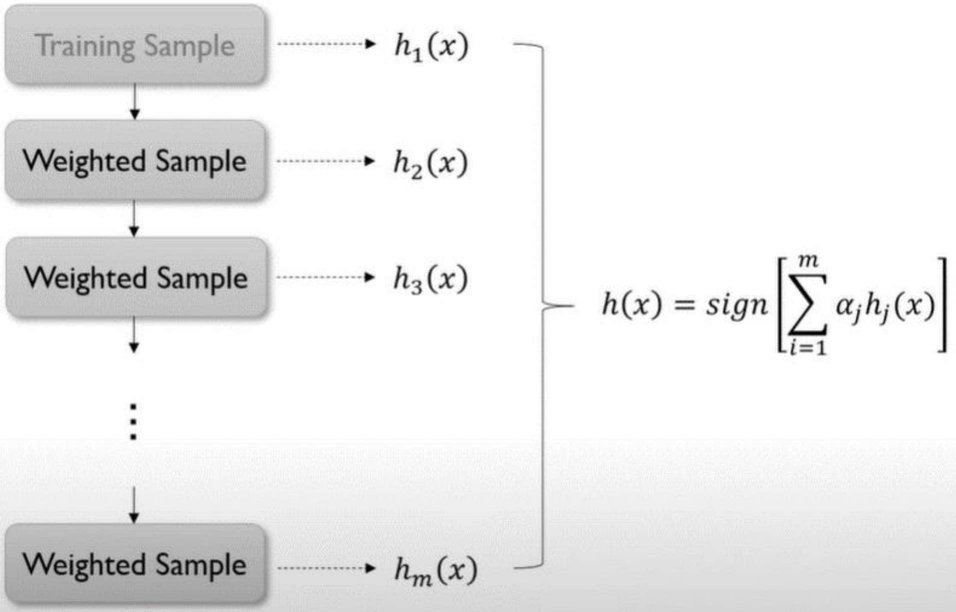
Step 2: Define the weight of a classifier: $\alpha_j = \log\left(\frac{1-L_j}{L_j}\right)$

Step 3: Update weight: $W_i \leftarrow W_i\, e^{\alpha_j I(y_i \neq h_j(x))}, i = 1,2,\dots,n$

endfor

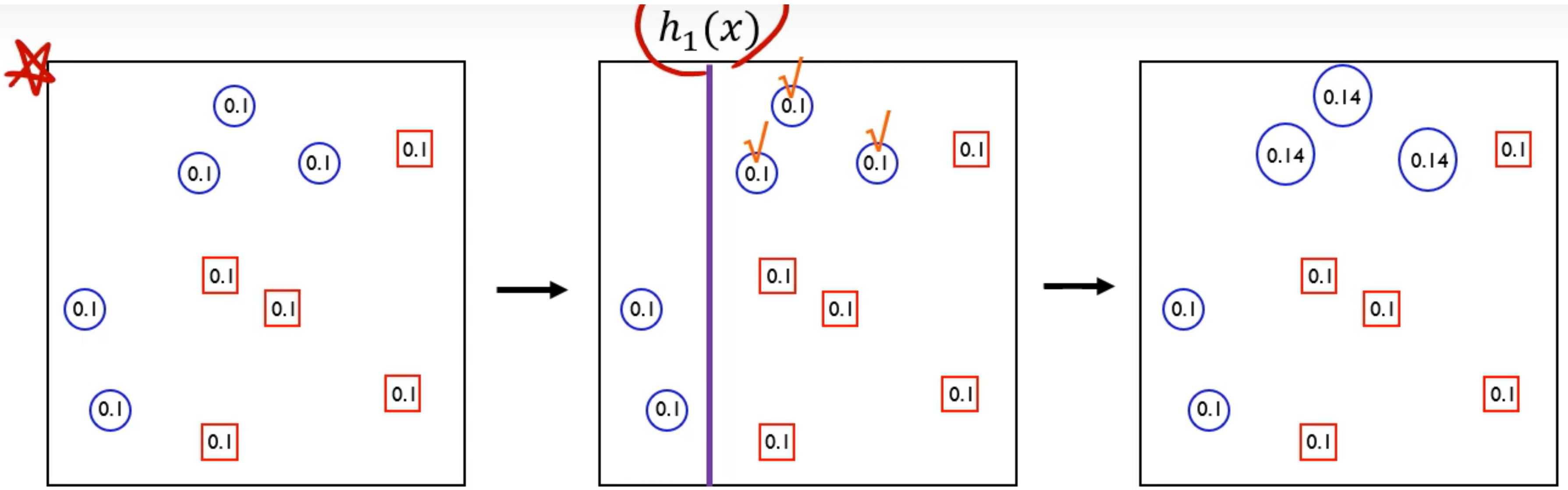
3. Final boosted model: $h(x) = sign\left[\sum_{i=1}^{m} \alpha_j h_j(x)\right]$

AdaBoost

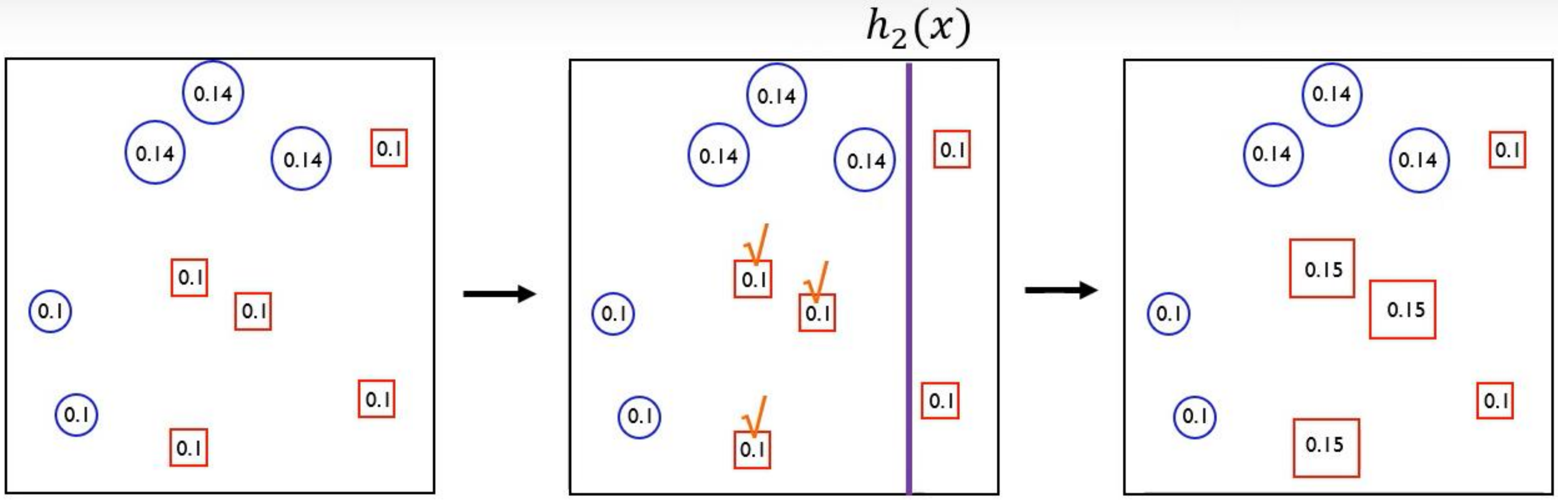


$h_1(x)$를 만들고, 이를 바탕으로, $h_2(x)$를 만들고, 이를 바탕으로 $h_3(x)$, ...

AdaBoost

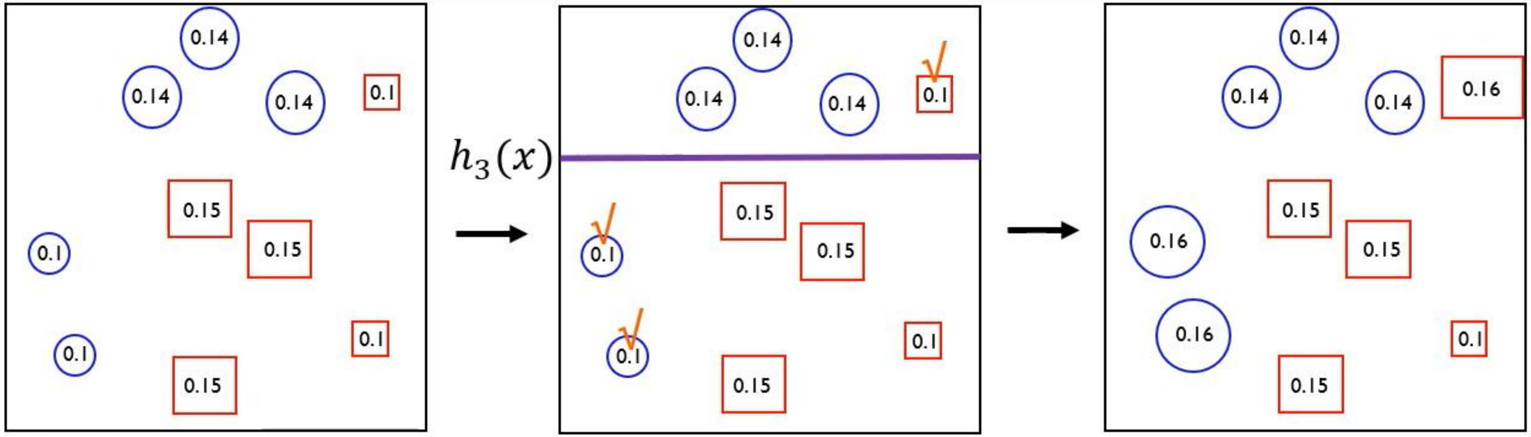


AdaBoost



$h_2(x)$
0.14
0.14
0.14
0.1
0.1
0.1
0.1
0.1
0.1
0.1
0.14
0.14
0.14
0.1
0.1
0.1
0.1
0.1
0.1
0.1
0.14
0.14
0.14
0.1
0.15
0.15
0.1
0.1
0.1
0.15

AdaBoost



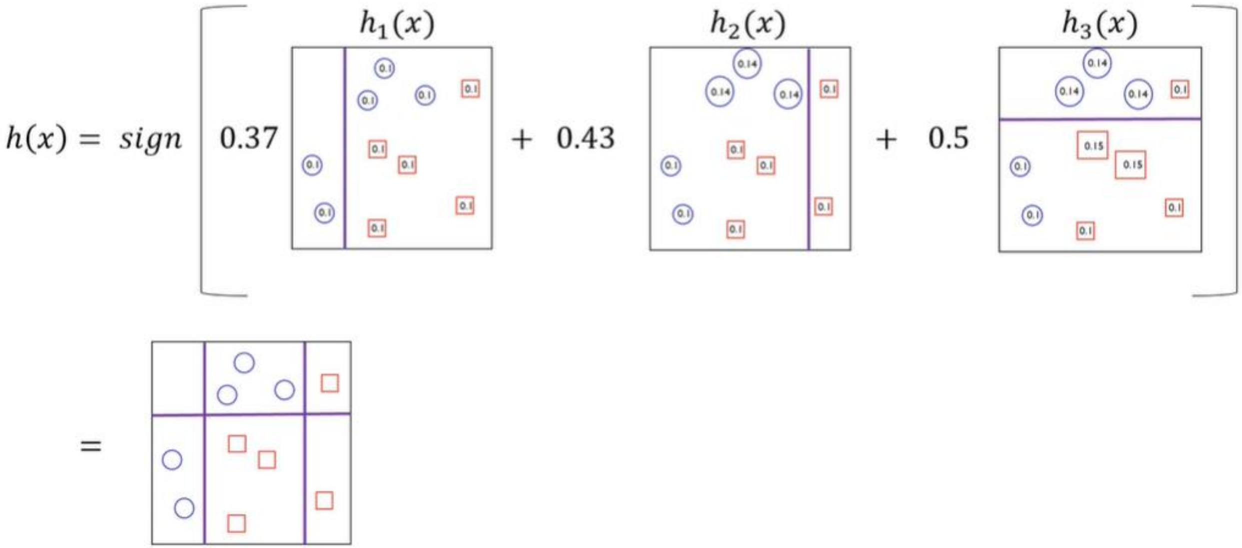
$$L_3 = \frac{\sum_{i=1}^{n} W_i \, I(y_i \neq h_2(x))}{\sum_{i=1}^{n} W_i} = \frac{0.1 \times 3}{0.1 \times 4 + 0.14 \times 3 + 0.15 \times 3} = 0.24$$

10개중 3개 오분류

$$\alpha_3 = \log\left(\frac{1 - 0.24}{0.24}\right) \approx 0.5$$

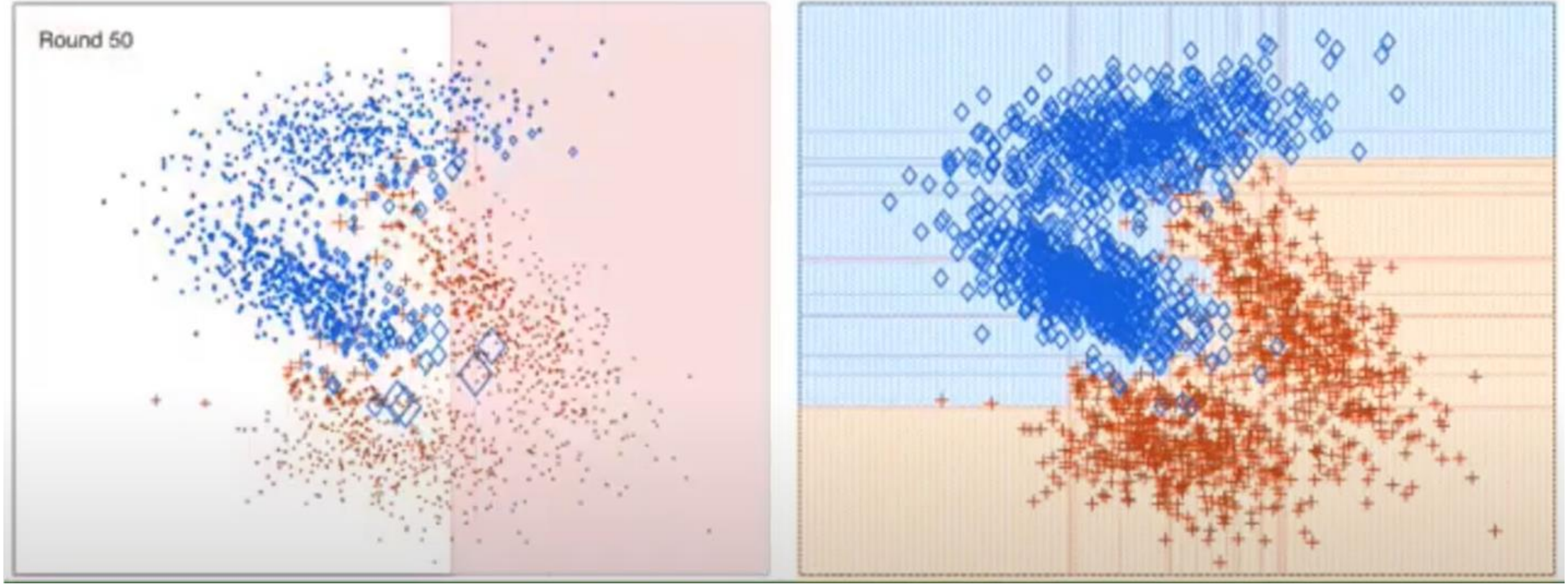
AdaBoost

$$h(x) = sign\left[\sum_{i=1}^{m=3} \alpha_j h_j(x)\right]$$



- 순차적 학습
- 오답 노트

AdaBoost



Coding Session

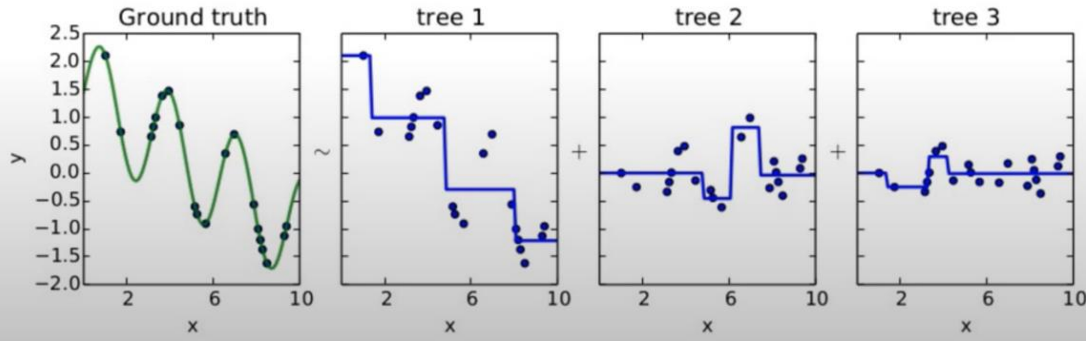
AdaBoost

- base_estimator: 학습할 때 사용할 estimator 종류 (Tree Model이 default)
- n_estimators: 복원 추출 횟수
- learning_rate: 가중치의 갱신 변동폭
- algorithm: 손실함수 종류

Gradient Boosting Machine (GBM)

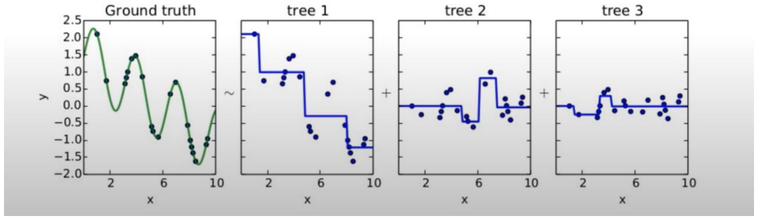
Gradient Boosting Machine (GBM)

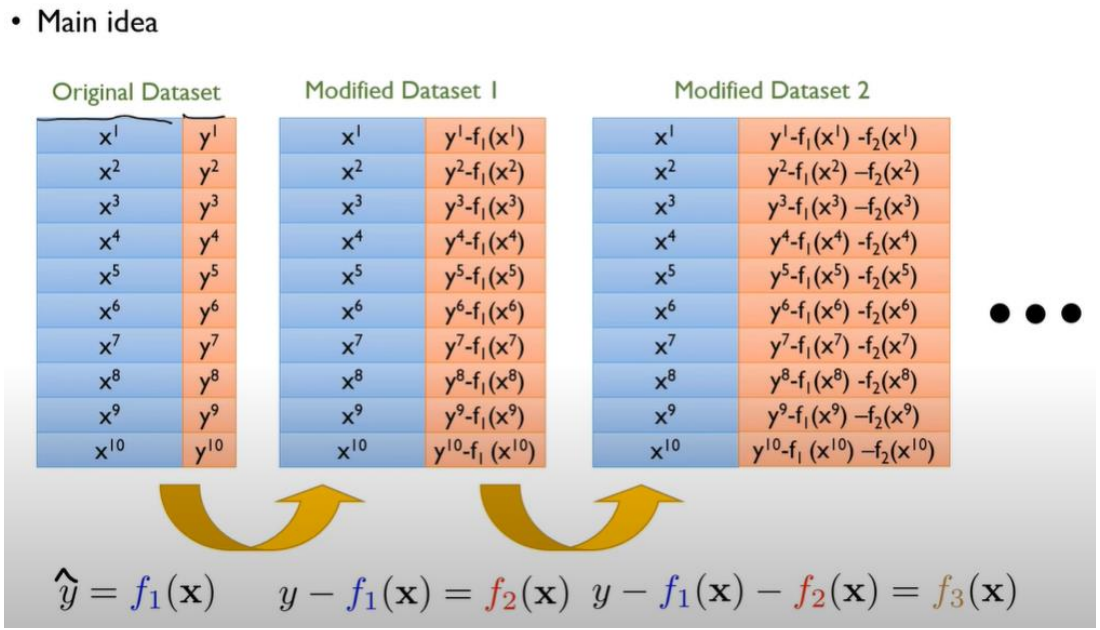
- Gradient boosting = Boosting with gradient decent
- 첫번째 단계의 모델 tree1을 통해 Y를 예측하고, Residual을 다시 두번째 단계 모델 tree2를 통해 예측하고, 여기서 발생한 Residual을 모델 tree3로 예측
- 점차 residual 작아 짐
- Gradient boosted model = tree1 + tree2 + tree3



Gradient Boosting Machine (GBM)

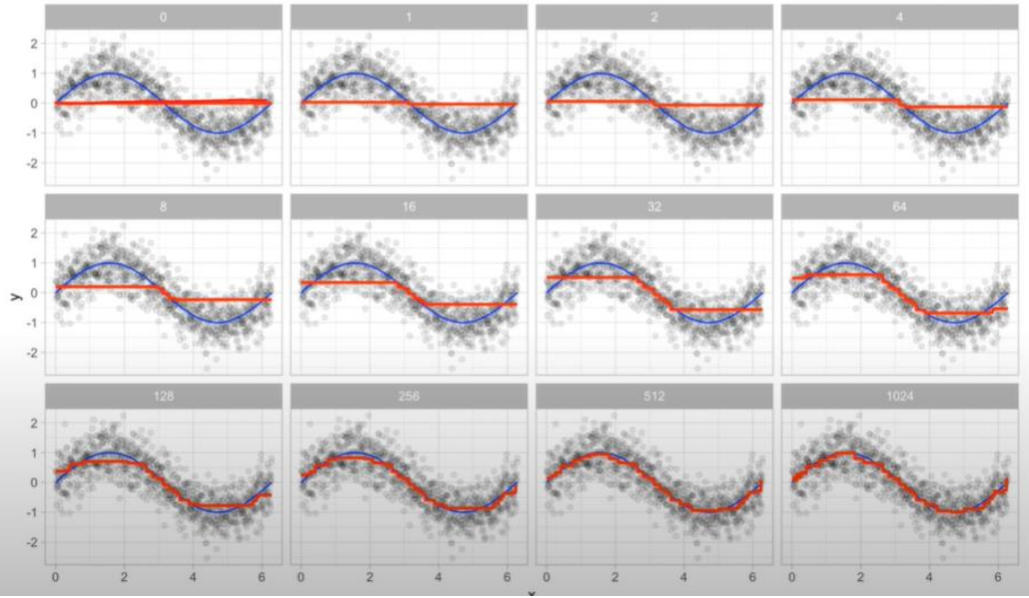
- Gradient boosting = Boosting with gradient decent
- 첫번째 단계의 모델 tree1을 통해 Y를 예측하고, Residual을 다시 두번째 단계 모델 tree2를 통해 예측하고, 여기서 발생한 Residual을 모델 tree3로 예측
- 점차 residual 작아 짐
- Gradient boosted model = tree1 + tree2 + tree3





Gradient Boosting Machine (GBM)

- GBM Regression Example 3



Gradient Boosting Machine (GBM)

- How is this idea related to the gradient?
  - ✓ Loss function of the ordinary least square (OLS)

$$\min L = \frac{1}{2}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2$$

  - ✓ Gradient of the Loss function

$$\frac{\partial L}{\partial f(\mathbf{x}_i)} = f(\mathbf{x}_i) - y_i$$

  - ✓ Residuals are the negative gradient of the loss function

$$y_i - f(\mathbf{x}_i) = -\frac{\partial L}{\partial f(\mathbf{x}_i)}$$

Gradient Boosting Machine (GBM)

- Gradient Boosting: Algorithm

  1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(y_i, \gamma)$.
  2. For $m = 1$ to M:

     2.1 For $i = 1, \ldots, N$ compute

     $$g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}$$

     2.2 Fit a regression tree to the targets g_{im} giving terminal regions $R_{jm}, j = 1, \ldots, J_m$.

     2.3 For $j = 1, \ldots, J_m$ compute

     $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

     2.4 Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

  3. Output $\hat{f}(x) = f_M(x)$.

Gradient Boosting Machine (GBM)

장점

- 최근 유명한 모델들 모두 GBM 기반 (성능 좋음)
- Variable Importance 계산 가능

✓ Variable importance of Gradient boosting

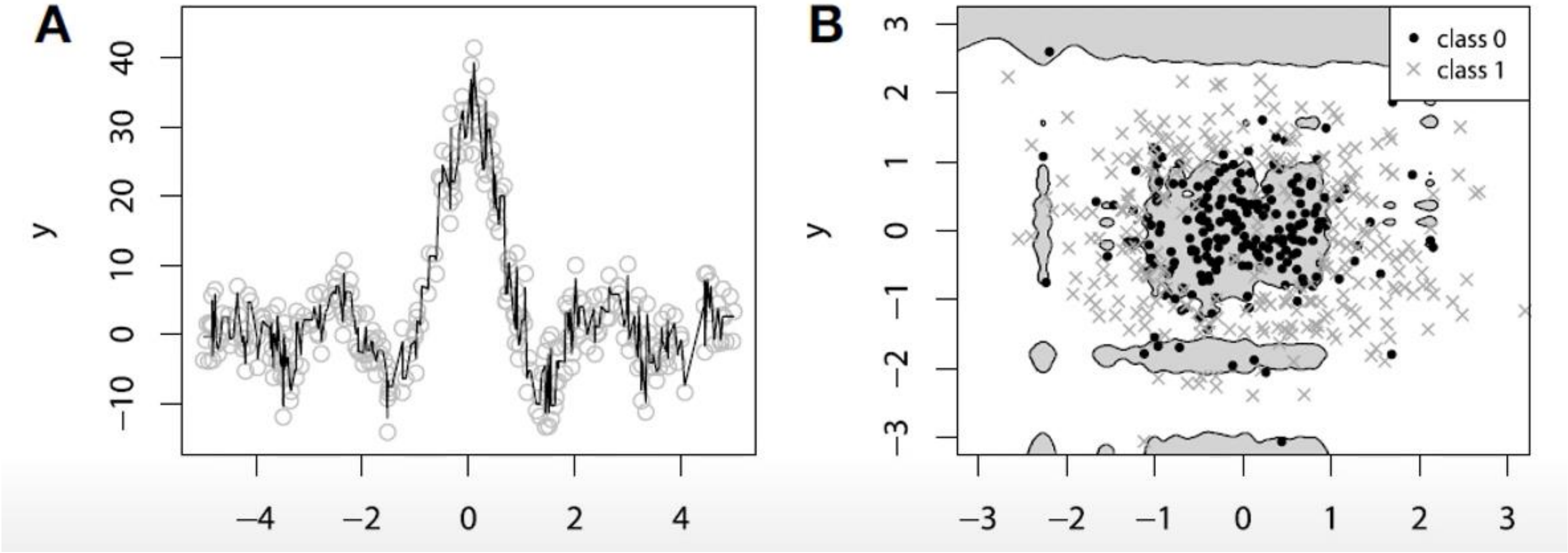
$$Influence_j = \frac{1}{M}\sum_{k=1}^{M} Influence_j(T_k)$$

단점

- Over Fitting 가능성
- 연산 속도가 오래 걸림

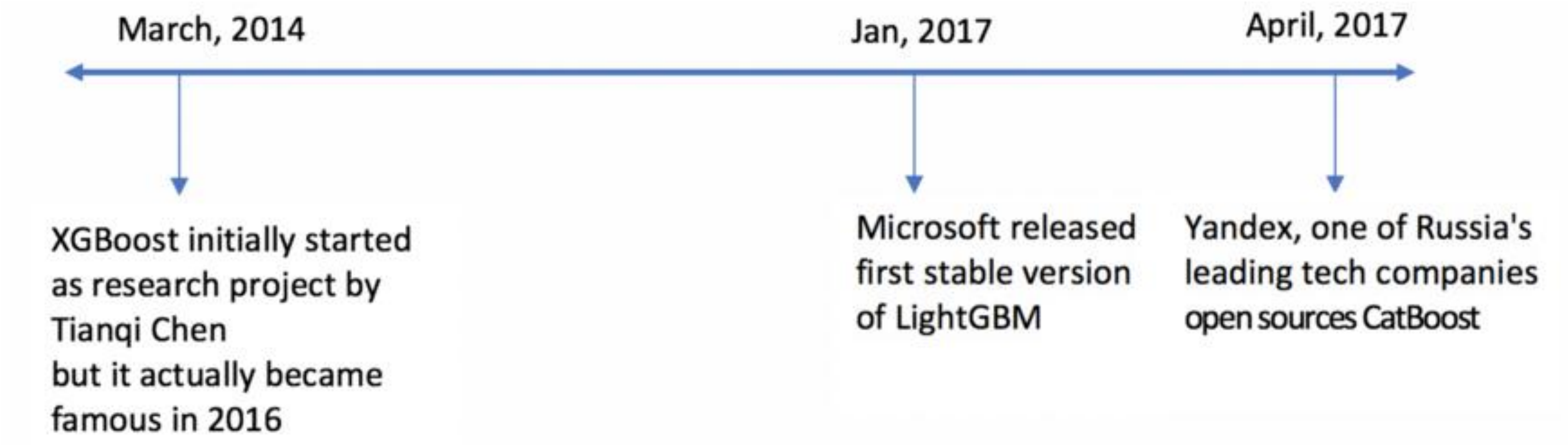
Gradient Boosting Machine (GBM)

Overfitting problem in GBM



Gradient Boosting Machine (GBM)

Overcoming GBM



Coding Session

GBM

sklearn.ensemble.GradientBoostingRegressor

class sklearn.ensemble.**GradientBoostingRegressor**(**, loss='squared_error', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, init=None, random_state=None, max_features=None, alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False, validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0*) [source]

- n_estimators: 복원 추출 횟수
- loss: 손실함수 종류
- subsample: 개별 트리가 학습에 사용하는 샘플링 비율

+Early Stopping

XgBoost

Split Finding Algorithm

Sparsity-Aware Split Algorithm

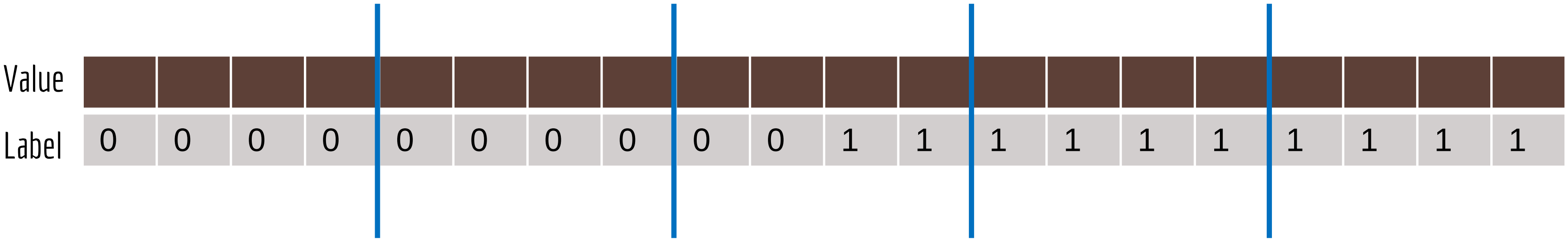
XgBoost

Previous Tree Models - Basic exact greedy algorithm

Value																				
Label	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

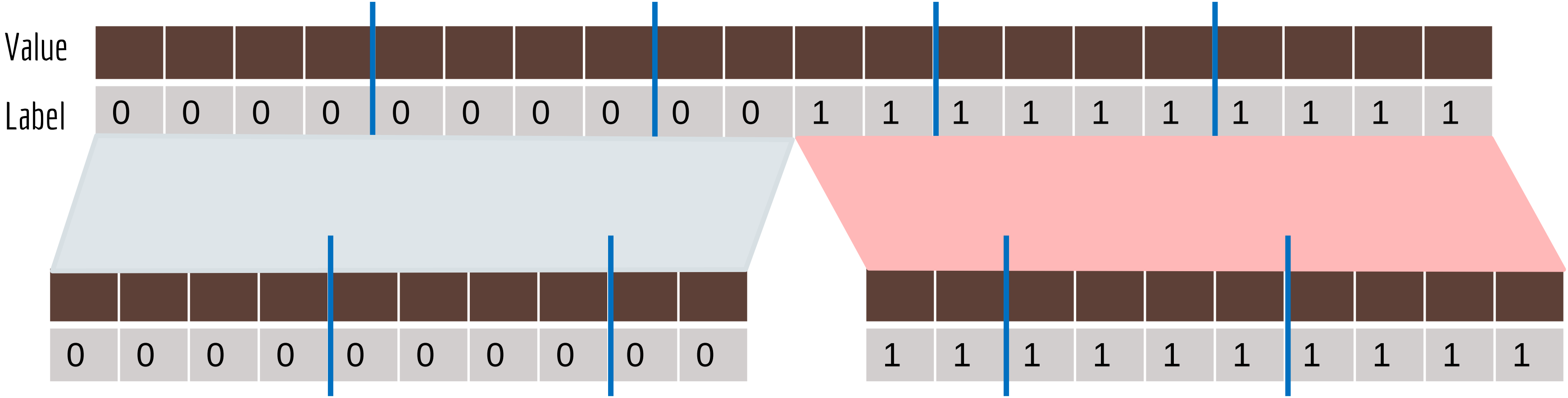
XgBoost

Split Finding Algorithm



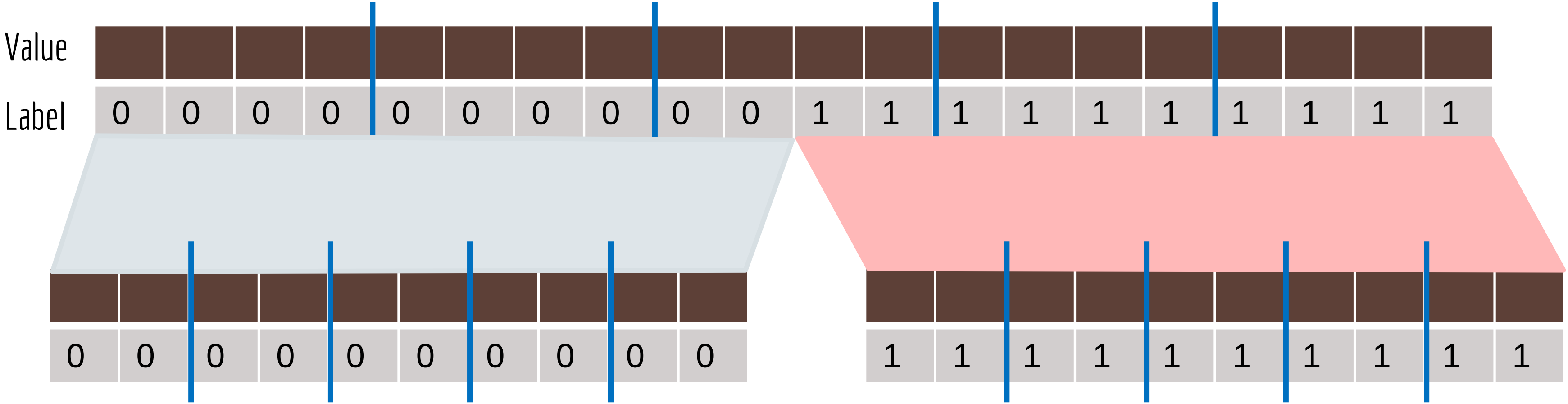
XgBoost

Global Variant



XgBoost

Local Variant



XgBoost

Algorithm 2: Approximate Algorithm for Split Finding

for $k = 1$ ***to*** m **do**

  Propose $S_k = \{s_{k1}, s_{k2}, \cdots s_{kl}\}$ by percentiles on feature k.

  Proposal can be done per tree (global), or per split(local).

end

for $k = 1$ ***to*** m **do**

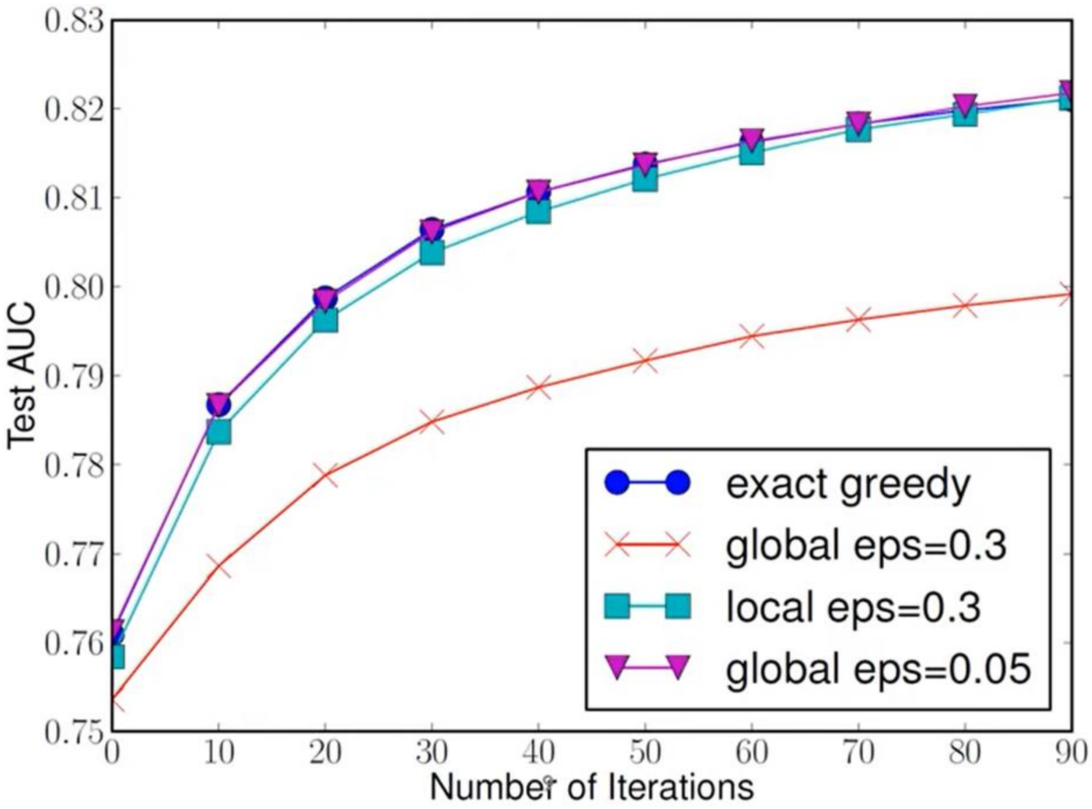
  $G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$

  $H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$

end

Follow same step as in previous section to find max score only among proposed splits.

XgBoost



Number of Buckets: 1/ξ

XgBoost

Sparsity-Aware Split Algorithm

Value	1.8		1.1	0.7		2.3	0.3		1.5	3.3	0.2	0.9
Label	1	0	1	0	0	01	0	0	1	1	0	0

Value	0.2		0.3	0.7		0.9	1.1		1.5	1.8	2.3	3.3
Label	0	0	0	0	0	0	1	0	1	1	1	1

Value	0.2	0.3	0.7	0.9	1.1	1.5	1.8	2.3	3.3			
Label	0	0	0	0	1	1	1	1	1	0	0	0

XgBoost

Sparsity-Aware Split Algorithm

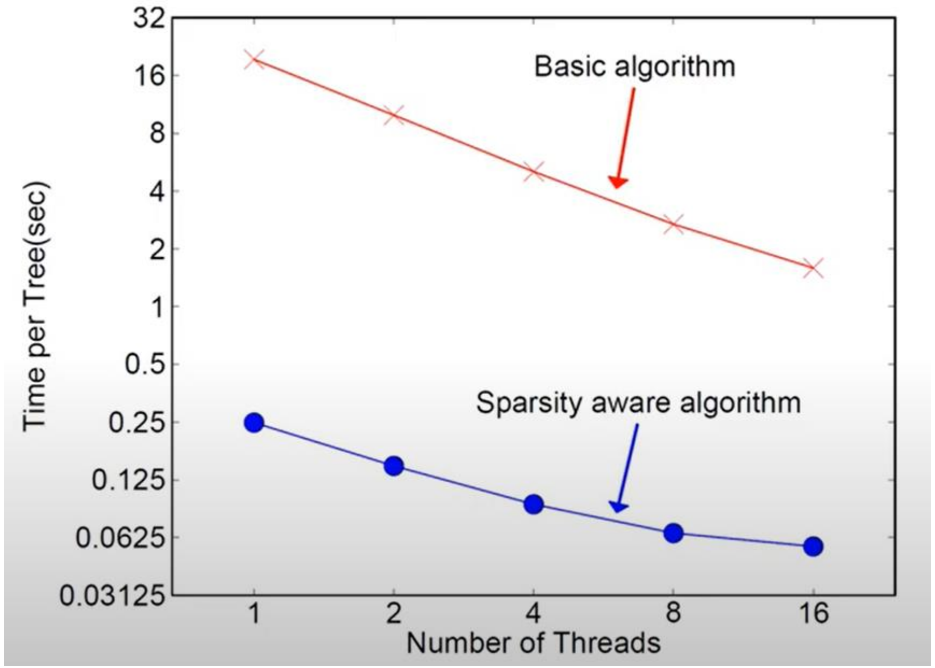
Value	1.8		1.1	0.7		2.3	0.3		1.5	3.3	0.2	0.9
Label	1	0	1	0	0	01	0	0	1	1	0	0

Value	0.2		0.3	0.7		0.9	1.1		1.5	1.8	2.3	3.3
Label	0	0	0	0	0	0	1	0	1	1	1	1

Value				0.2	0.3	0.7	0.9	1.1	1.5	1.8	2.3	3.3
Label	0	0	0	0	0	0	0	1	1	1	1	1

XgBoost

Sparsity-Aware Split Algorithm



XgBoost

자료 구조 및 하드웨어 처리

과적합 규제

Coding Session

XgBoost

class **xgboost.XGBClassifier**(*, objective='binary:logistic', use_label_encoder=False, **kwargs)

Bases: `xgboost.sklearn.XGBModel`, `sklearn.base.ClassifierMixin`

- **eta:** learning rate
- **min_child_weight:** 한 리프 노드에 필요한 인스턴스의 가중치 합
- **tree_method (exact, approx, hist, gpu hist):** 트리 분할 방법론