Machine Learning 2

Boosting, Stacking

Dept. SW and Communication Engineering

Prof. Giseop Noh (kafa46@hongik.ac.kr)

Contents

Recap: Bagging, RandomForest

Bagging: Bootstrap

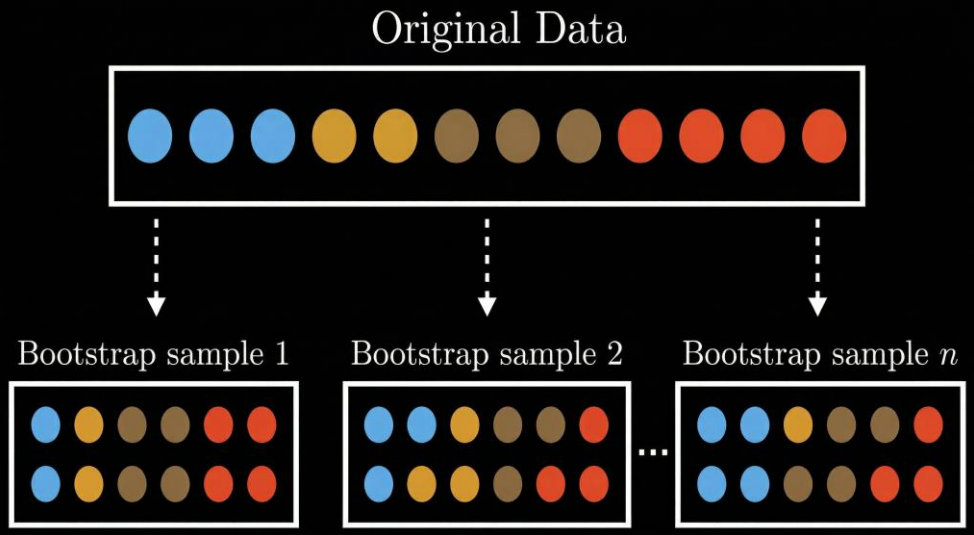
Bootstrap

Sampling with replacement

The same data point can be selected multiple times

Repeat:

Select one sample → Put it back → Select again → Put it back ⋯

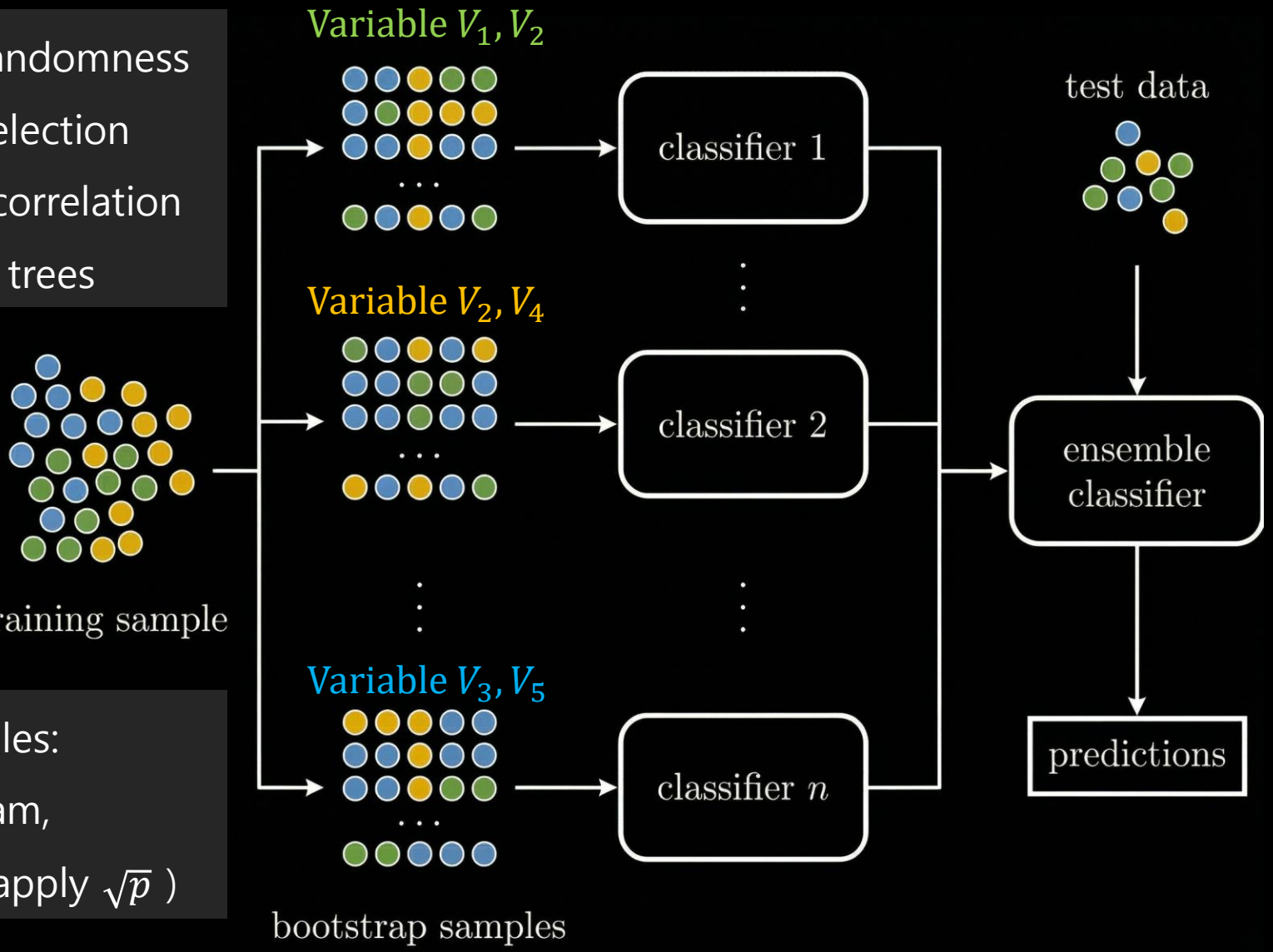


Property:

- Some data points are not selected in each sample on average: 36%

Random Selection of Variables

Introduce randomness
in feature selection
→ reduces correlation
between trees



# of variables:
Hyper param,
generally, apply $\sqrt{p}$)

Boosting

Taxonomy of Ensemble Learning

Pure Tree-based Approach

- Same data → same result
- Combining multiple tree models
- → still the same result
- No benefit from Ensemble Learning

Solution: Variety

Bagging

Data Reconfiguration (Bootstrap)

Build Models with Variety

RandomForest

Data Reconfiguration (Bootstrap)

+ Variables Reconfiguration

Boosting

Focus on wrong answers:

Enhance Prediction Scores

- AdaBoost
- GradientBoosting:
  - XgBoost, LightGBM, CatBoost

Boosting Types

Boosting: Focus on Difficult Problems (wrong answers)

- ✓ AdaBoost: Basic Approach
- ✓ LPBoost
- ✓ TotalBoost
- ✓ BrownBoost
- ✓ MadaBoost
- ✓ LogitBoost

⋮

- ✓ GradientBoost (XgBoost, LightGBM, CatBoost):

  → Widely used Boosting Approaches

We will cover!

Basic Principle of Boosting

성적 향상을 위해
오답 노트를 이용하자!

- Assign higher weights to misclassified data
- Initially, all data have equal weights
  → as rounds proceed,
  data weights and
  learner importance are updated
- Apply weights during sampling
  with replacement
- Misclassified data receive higher weights
  → more likely to be sampled
  in the next round

AdaBoost (Adaptive Boosting)

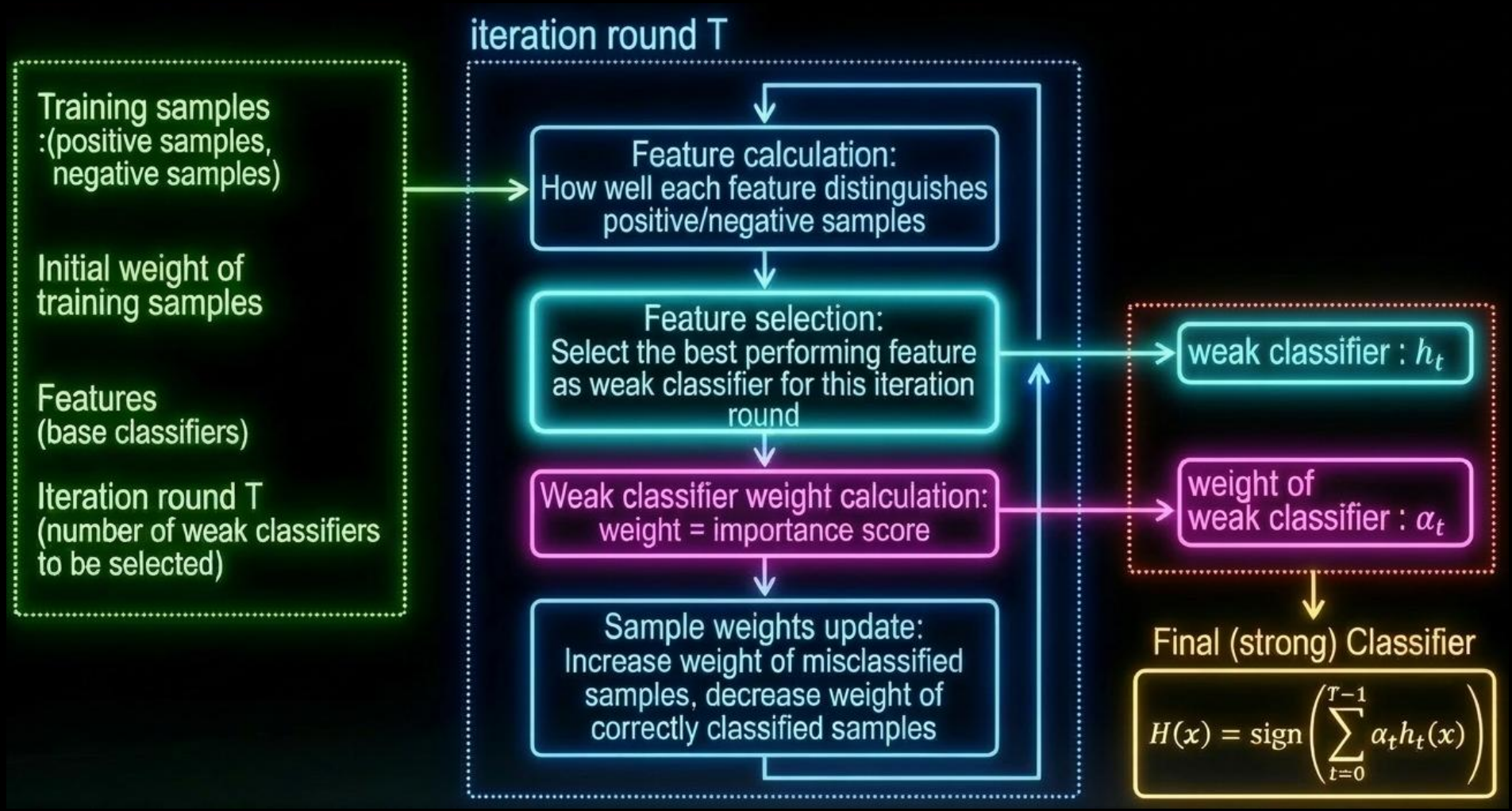
Assign higher weights to misclassified data

Update weights over multiple rounds

→ perform weighted sampling during bootstrap

(sampling with replacement)

AdaBoost Mechanism



Exercise

Practice:

AdaBoost Practice with Codes

Gradient Boosting: XGBoost

Boosting Idea

Boosting Idea

- AdaBoost: Assign higher weights to misclassified samples
- Gradient Boosting: Train a new weak learner to predict the residual error of the previous model

Gradient Boosting Steps:

$$y = h_0(x) + \epsilon_0$$

x: Input

ϵ_0: error

y: Target

$h_0(x)$: Learner

Types of Error:

Irreducible Error:

Cannot be reduced; unavoidable

Reducible Error:

Can be reduced through learning (training)

Residual Learning

Reducible Error:

Can be reduced through learning (training)

$$y = h_0(x) + \epsilon_0 \rightarrow \hat{y} = h_0(x)$$

당연히 틀림
(오차 존재)

Iterative Residual Learning

Compute residual: $r_m = y - \hat{y}_m$

Train next model: $h_{m+1}(x) \approx r_m$

Update prediction: $\hat{y}_{m+1} = \hat{y}_m + h_{m+1}(x)$

Final Model: $\hat{y} = h_0(x) + \eta \sum_{m=1}^{M} h_m(x)$

$h_0(x)$: initial prediction (e.g., mean of y)

$h_m(x)$: residual correction term (weak learners)

η: learning rate

$$r_0 = y - h_0(x)$$

$$h_1(x) \approx r_0$$

$$r_1 = r_0 - h_1(x)$$

$$h_2(x) \approx r_1$$

$$\vdots$$

$$r_{m+1} = r_m - h_{m+1}(x)$$

$$h_{m+1}(x) \approx r_m$$

틀린 부분만
따로 뽑음

틀린 부분만
맞추려고 노력

계속 틀린 부분만 수정
→ 점점 오차 감소

Gradient Boosting

Objective: $\min_{\hat{y}} L(y, \hat{y})$

Core Idea: $h_{m+1}(x) \approx -\frac{\partial L(y, \hat{y})}{\partial \hat{y}_m}$

$$L = \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y$$

$$-\frac{\partial L}{\partial \hat{y}} = y - \hat{y} = r_m$$

Move in the direction that reduces the loss most rapidly (gradient direction)

- The gradient indicates the direction to reduce error
- Add a model in that direction

Is it working with any problems?

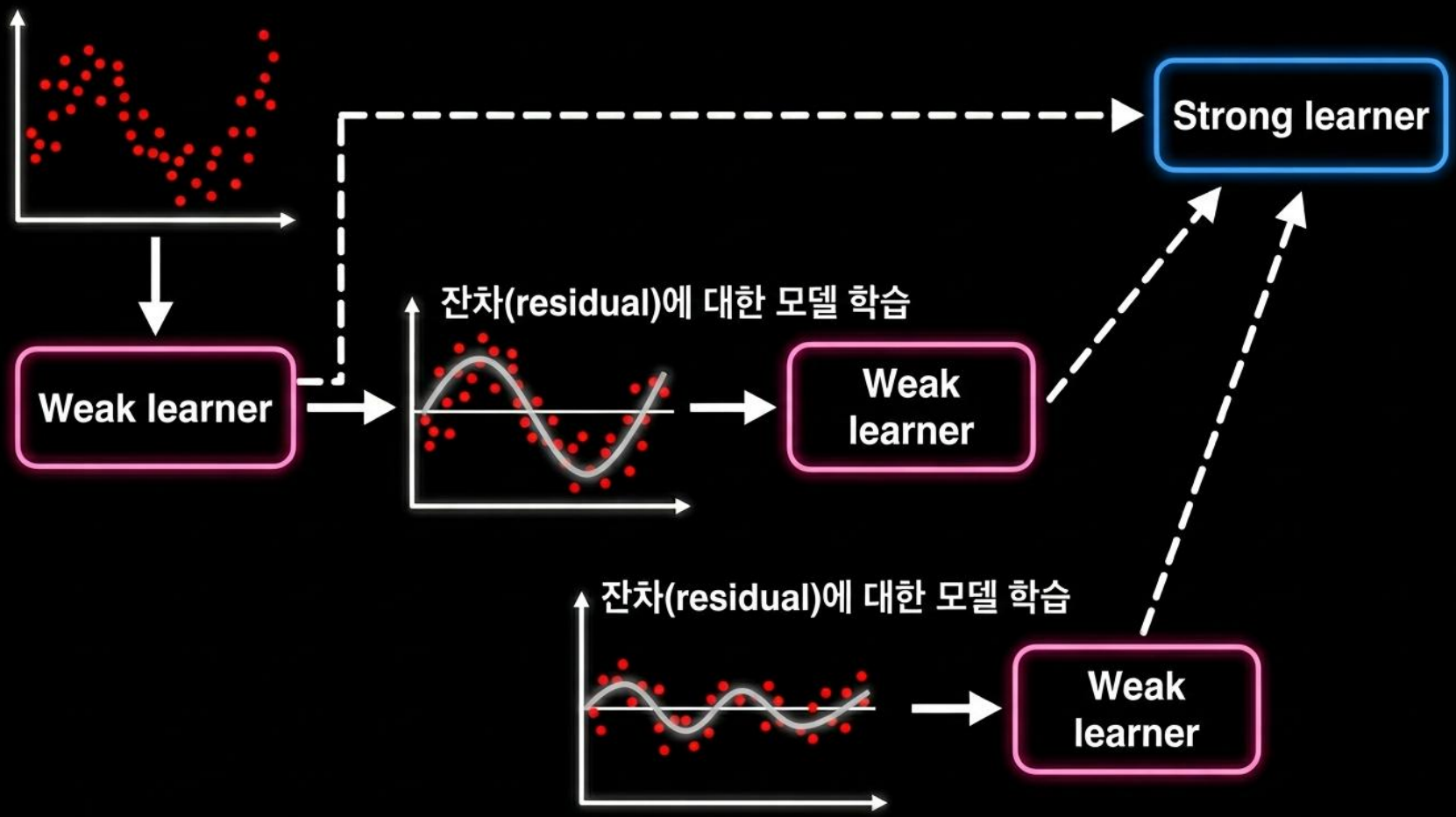
Regression Loss	Classification Loss
MSE	Cross Entropy
$L = \frac{1}{2}(y - \hat{y})^2$	$L = -\left[y \log p(x) + (1 - y) \log\left(1 - p(x)\right)\right]$
$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y$	$\frac{\partial L}{\partial F(x)} = p(x) - y$
$-\frac{\partial L}{\partial \hat{y}} = y - \hat{y} = r_m$	$-\frac{\partial L}{\partial F(x)} = y - p(x)$
	$h_{m+1}(x) \approx y - p(x)$

$F(x)$: logit

$p(x) = \sigma\left(F(x)\right) = \frac{1}{1 + e^{-F(x)}}$

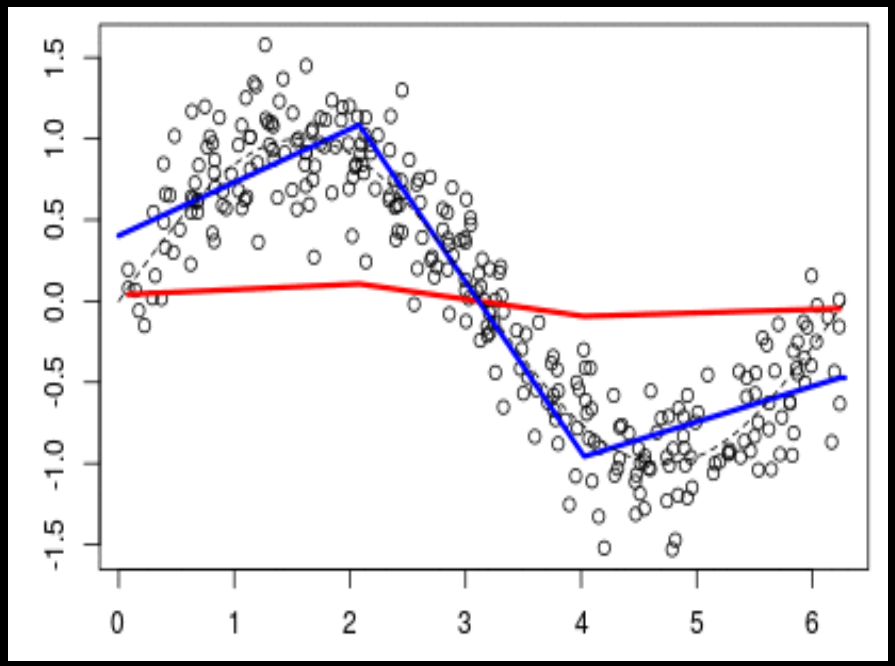
$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial p} \cdot \frac{\partial p}{\partial F}$

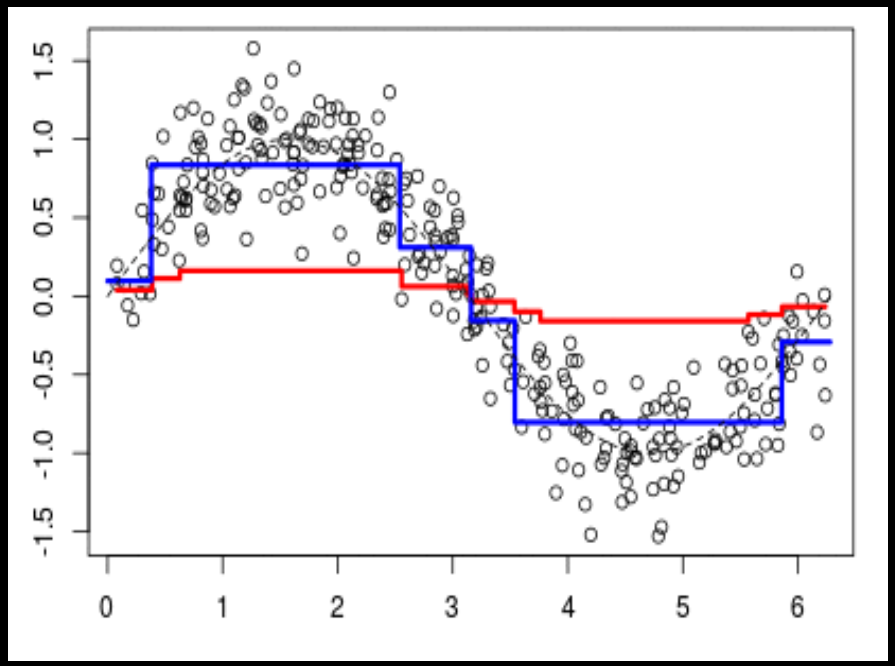
Visual Understanding (1/2)



Visual Understanding (2/2)

- 검은 점: 실제 데이터
- 파란 선: 현재 모델 (strong learner)
- 빨간 선: residual (오차, $y - \hat{y}$)





Source: https://www.r-bloggers.com/2015/06/an-attempt-to-understand-boosting-algorithms/

Gradient Boosting: XGBoost (1/2)

XGBoost (eXtreme Gradient Boosting)

It's known that winning teams in the Kaggle and KDD competitions used it a lot.

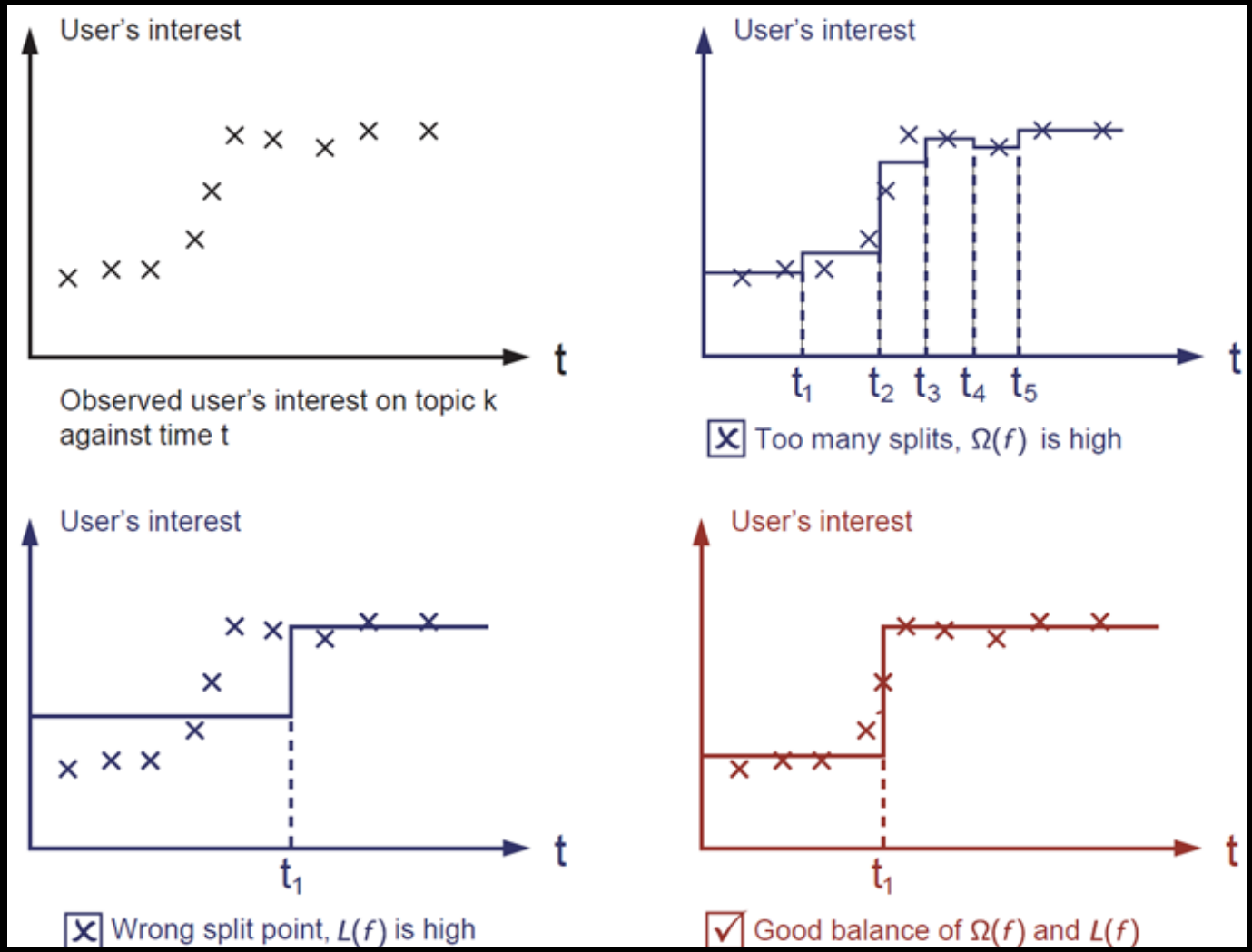
Very similar to traditional Gradient Boosting:

- Minimize Residual by applying data errors to the next round

Differences:

- Regularization items are added to learning objectives to prevent overfitting

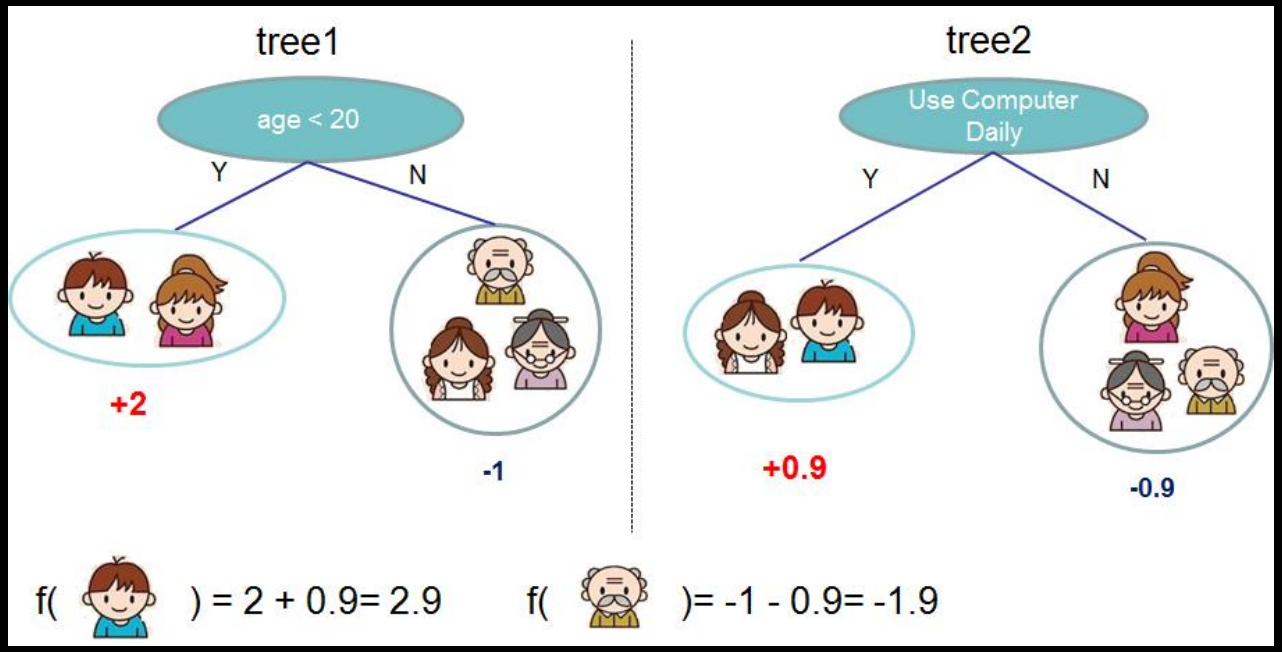
Gradient Boosting: XGBoost (2/2)



Source: https://xgboost.readthedocs.io/en/latest/tutorials/model.html

XGBoost Concept (1/4)

Dataset: $D = \{(x_i, y_i)\}_{i=1}^{n}, x_i \in \mathbb{R}^m, y_i \in \mathbb{R}$



$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F \qquad F = \{f(x) = w_{q(x)}\}, where\ q: \mathbb{R}^m \to T, w \in \mathbb{R}^T$$

T: # of leaves in tree

XGBoost Concept (2/4)

$obj^t(\theta) = \sum_i^n l(y_i, \hat{y}_i) + \sum_{k=1}^t \omega(f_k)$, where $\omega(f_k)$: the complexity of the tree f_k

$obj^t(\theta) = \sum_i^n l(y_i, \hat{y}_i^t) + \sum_{k=1}^t \omega(f_k)$

t: the number of trees in ensemble
Each training step will add one new tree, so that at step t the ensemble contains $K = t$ trees

$\hat{y}_i^0 = 0$

$\hat{y}_i^1 = f_1(x_i) = \hat{y}_i^0 + f_1(x_i)$

$\hat{y}_i^2 = f_1(x_i) + f_2(x_i) = \hat{y}_i^1 + f_2(x_i)$

…

$\hat{y}_i^t = \sum_{k=1}^t f_k(x_i) = \hat{y}_i^{t-1} + f_t(x_i)$

$obj^t(\theta) = \sum_i^n \left(y_i - \hat{y}_i^{t-1} + f_t(x_i)\right)^2 + \sum_{k=1}^t \omega(f_k)$

XGBoost Concept (3/4)

$$obj^t(\theta) = \sum_i^n \left(y_i - \hat{y}_i^{t-1} + f_t(x_i)\right)^2 + \sum_{k=1}^t \omega(f_k)$$

Approximation by Tylor Series

$$obj^t(\theta) = \sum_i^n \left[l\left(y_i, \hat{y}_i^t\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \omega(f_k) + constant$$

$$g_i = \frac{\partial l\left(y_i, \hat{y}_i^{t-1}\right)}{\partial \hat{y}_i^{t-1}} \qquad h_i = \frac{\partial^2 l\left(y_i, \hat{y}_i^{t-1}\right)}{\partial \left(\hat{y}_i^{t-1}\right)^2}$$

Removing constants

$$obj^t(\theta) = \sum_i^n \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \omega(f_k)$$

$$\omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^T w_j^2$$

w: the vector of scores on leaves

q: a function assigning each data point to the corresponding leaf

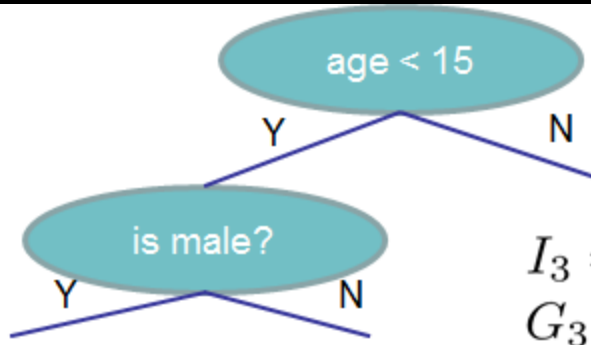
T: the number of trees

$$\omega_j^* = -\frac{G_j}{H_j + \lambda}$$

$$obj^* = -\frac{1}{2}\sum_{j=1}^T \frac{G_j}{H_j + \lambda} + \gamma T$$

XGBoost Concept (4/4)

Instance index		gradient statistics
1		g1, h1
2		g2, h2
3		g3, h3
4		g4, h4
5		g5, h5



$$I_1 = \{1\} \quad G_1 = g_1 \quad H_1 = h_1$$

$$I_2 = \{4\} \quad G_2 = g_4 \quad H_4 = h_4$$

$$I_3 = \{2, 3, 5\} \quad G_3 = g_2 + g_3 + g_5 \quad H_3 = h_2 + h_3 + h_5$$

$$Obj = -\sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

The smaller the score is, the better the structure is

XGBoost 참고 자료

Official Home: https://xgboost.readthedocs.io/en/latest/tutorials/model.html

Blog URL: https://soobarkbar.tistory.com/32

Paper Link: http://dmlc.cs.washington.edu/data/pdf/XGBoostArxiv.pdf

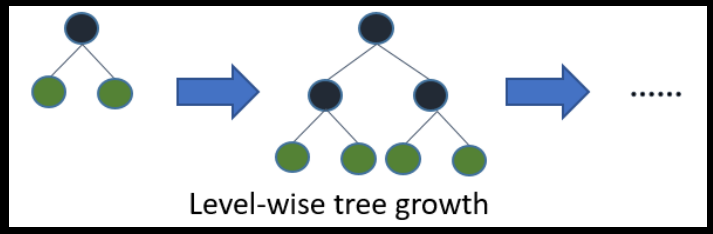
Tylor Series: https://darkpgmr.tistory.com/59

Gradient Boosting: LightBoost

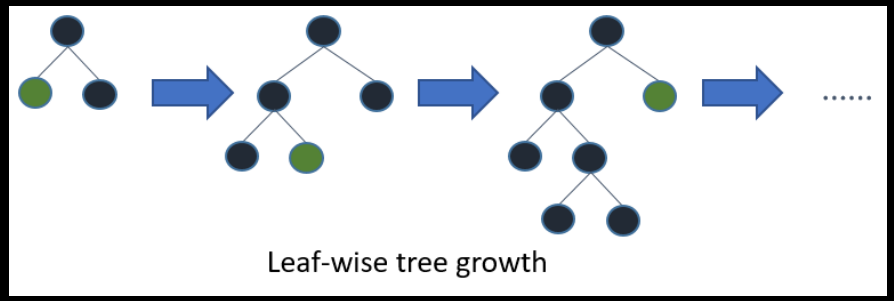
LightGBM

- Use Leaf-wise Loss
  - Can reduce loss more
  - More sensitive to overfitting instead
- Faster than XGBoost (Omit the tree leveling operation)
- GPU support
- Enable large amounts of learning data

XGBoost



LightGBM



Gradient Boosting: CatBoost

CatBoost (unbiased boosting with Categorical features)

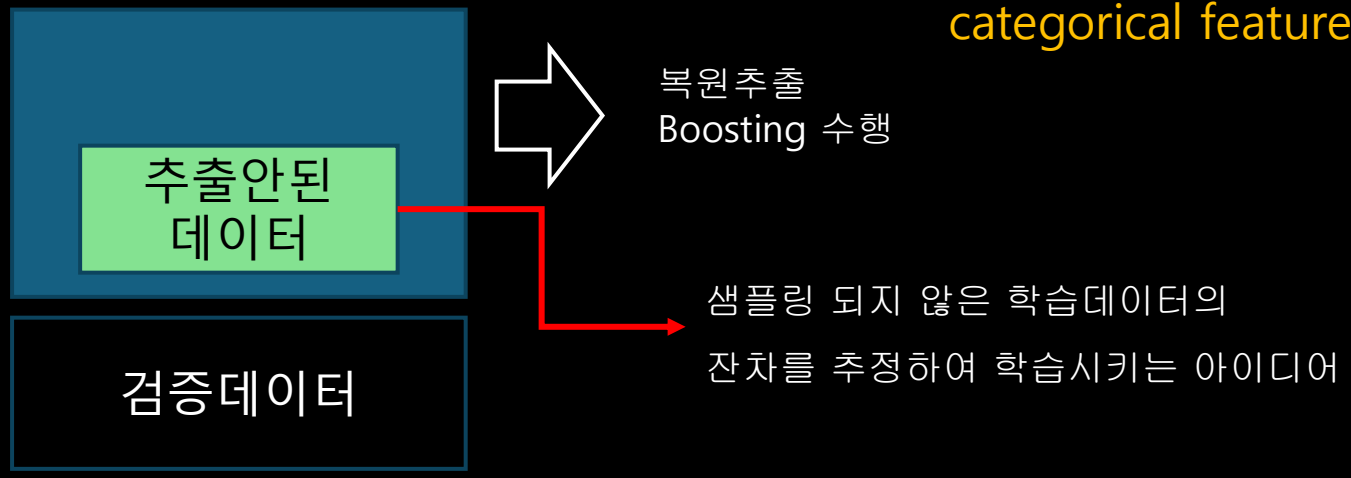
Key Idea:

- Reduces prediction shift (bias) in gradient boosting
- Achieves better generalization by controlling bias–variance trade-off

Core Mechanism:

- Uses ordered boosting
- Each sample is predicted using a model trained without that sample
- Enables unbiased residual estimation

Strong performance with categorical features



Exercise

Practice:

Gradient Boost Practice with Codes

- XGBoost
- LightGBM
- CatBoost

Stacking

Need of Stacking

Pure Tree-based Approach

- Same data → same result
- Combining multiple tree models
- → still the same result
- No benefit from Ensemble Learning

Solution: Variety

Bagging

Data Reconfiguration (Bootstrap)
Build Models with Variety

RandomForest

Data Reconfiguration (Bootstrap)
+ Variables Reconfiguration

Boosting

Focus on wrong answers:
Enhance Prediction Scores

- AdaBoost
- GradientBoosting:
  - XgBoost,
  - LightGBM,
  - CatBoost

Stacking

Model Output

→ Apply as New Independent Variables

Stacking

Stacking?

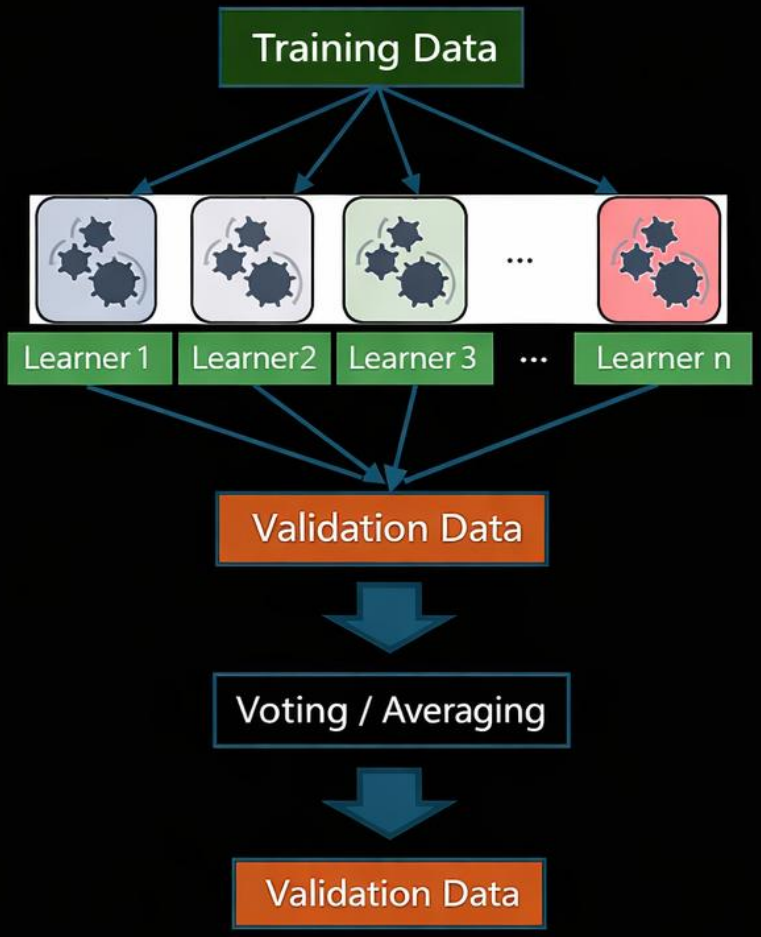
- A machine learning ensemble technique
- Combines multiple models by stacking them in layers
- Uses outputs of base models as new input features

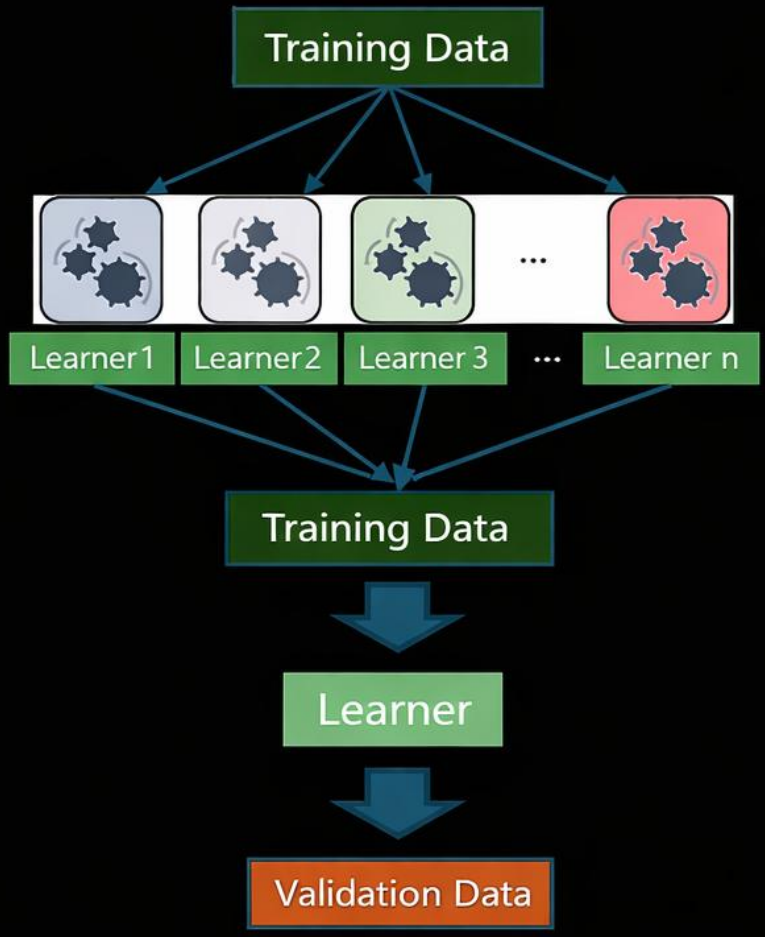
Core Idea

- Train multiple base learners (Level-0 models)
- Collect their predictions
- Train a meta-model (Level-1 model) on these predictions

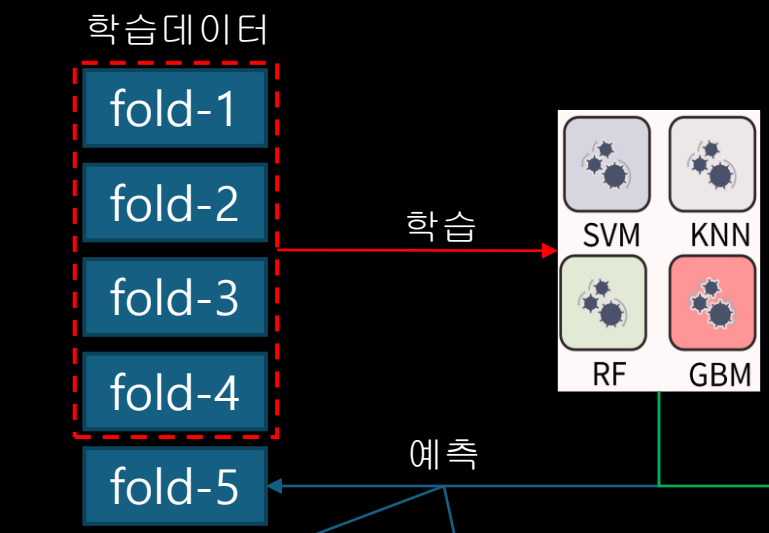
Differences from Other Ensemble Methods

Stacking: Combine Different Models





n-fold 교차검증 (cross validation) 1st - Round



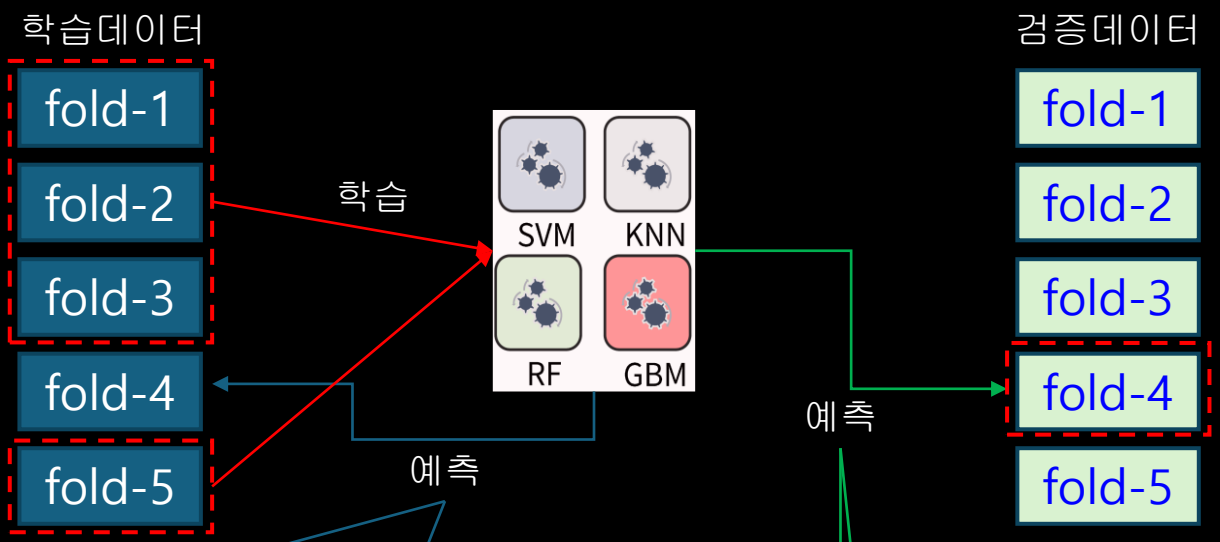
변수1	변수2
0.2	0.1
0.6	-0.3
-0.7	0.2

SVM	KNN	RF	GBM
불량	불량	불량	정상
정상	정상	불량	정상
정상	불량	불량	불량

변수1	변수2
0.4	0.3
0.3	-0.2
-0.7	0.4

SVM	KNN	RF	GBM
불량	정상	불량	정상
정상	정상	정상	정상
정상	불량	불량	불량

n-fold 교차검증 (cross validation) 2nd - Round



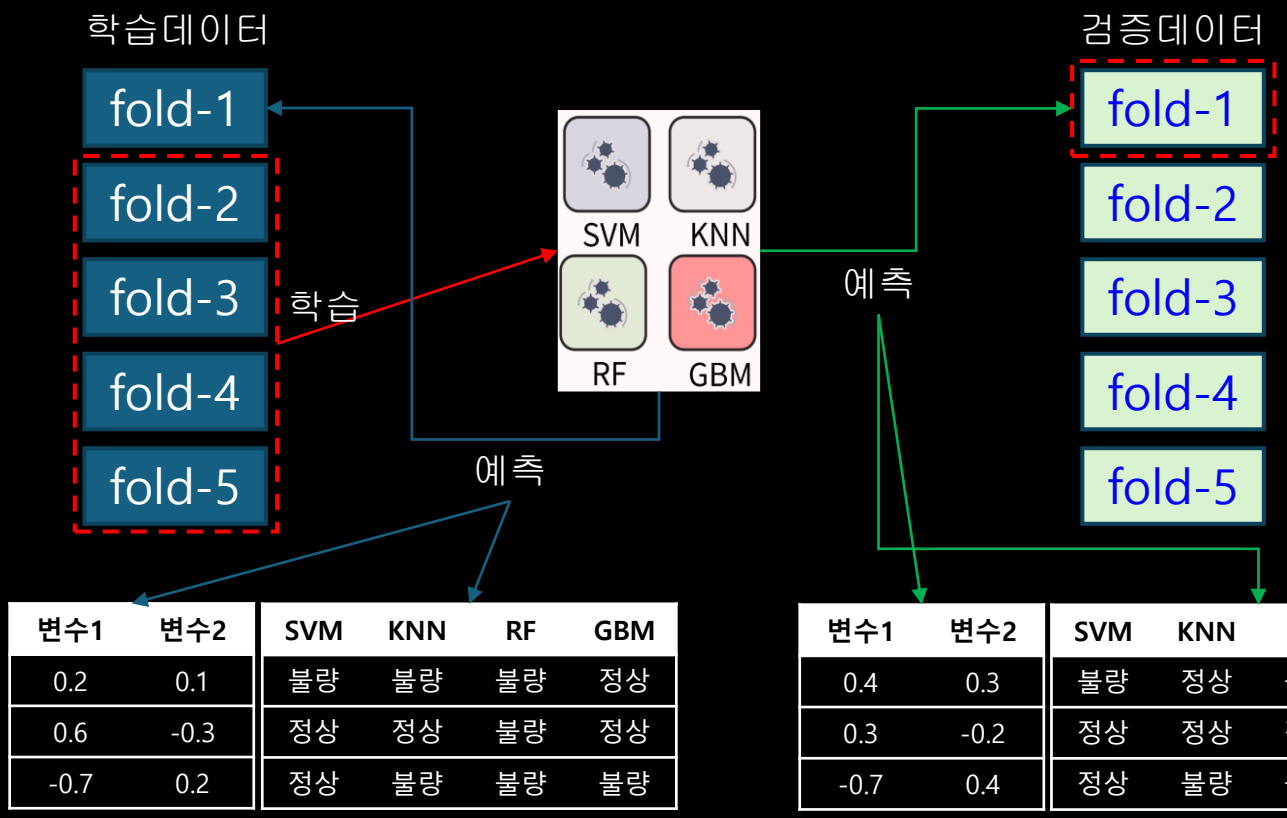
변수1	변수2
0.2	0.1
0.6	-0.3
-0.7	0.2

SVM	KNN	RF	GBM
불량	불량	불량	정상
정상	정상	불량	정상
정상	불량	불량	불량

변수1	변수2
0.4	0.3
0.3	-0.2
-0.7	0.4

SVM	KNN	RF	GBM
불량	정상	불량	정상
정상	정상	정상	정상
정상	불량	불량	불량

n-fold 교차검증 (cross validation) 5th - Round



변수1	변수2
0.2	0.1
0.6	-0.3
-0.7	0.2

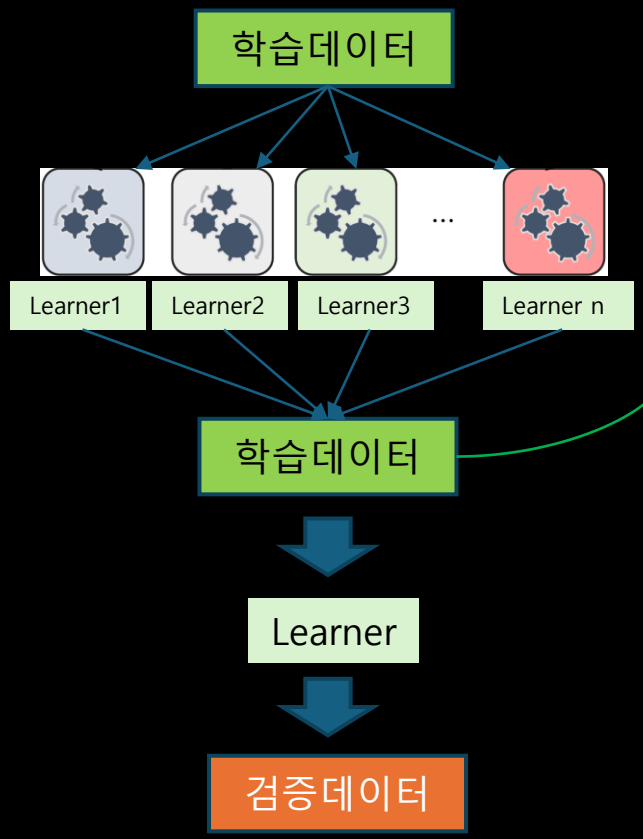
SVM	KNN	RF	GBM
불량	불량	불량	정상
정상	정상	불량	정상
정상	불량	불량	불량

변수1	변수2
0.4	0.3
0.3	-0.2
-0.7	0.4

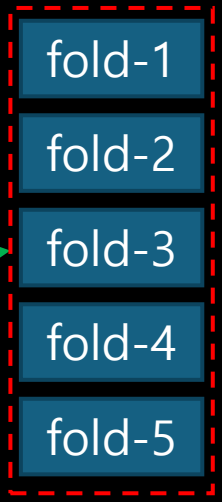
SVM	KNN	RF	GBM
불량	정상	불량	정상
정상	정상	정상	정상
정상	불량	불량	불량

Build New Train Dataset

Stacking: 다양한 모델을 결합



학습데이터



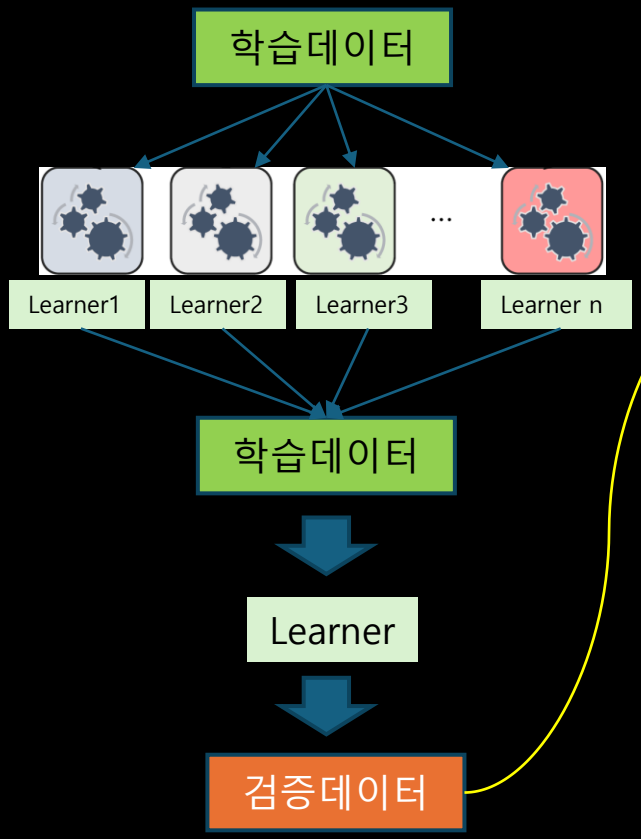
새로운 학습데이터

변수1	변수2	SVM	KNN	RF	GBM
0.2	0.1	불량	정상	불량	정상
0.6	-0.3	정상	정상	정상	정상
-0.7	0.2	정상	불량	불량	불량
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:

기존 학습데이터 각 모델별 예측값

Build New Validation Dataset

Stacking: 다양한 모델을 결합



학습데이터

- fold-1
- fold-2
- fold-3
- fold-4
- fold-5

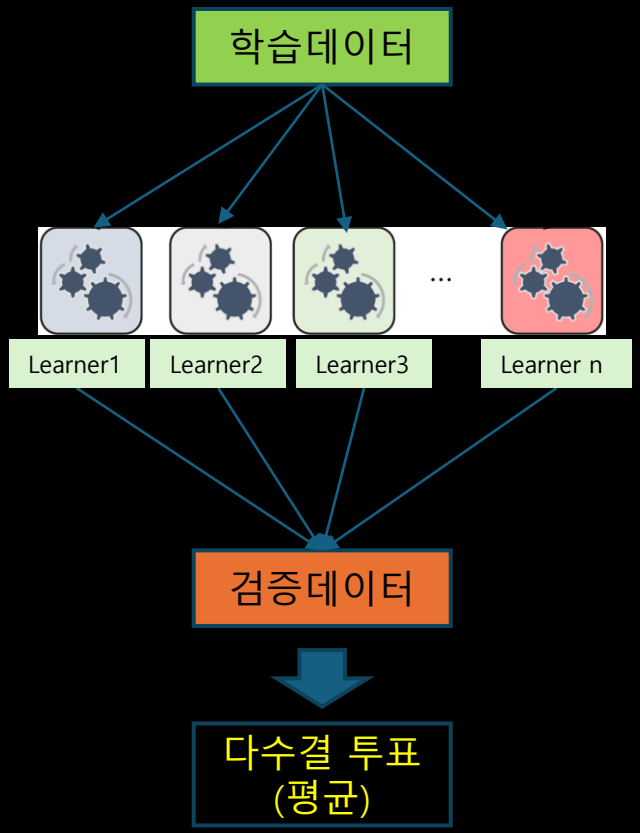
새로운 검증데이터

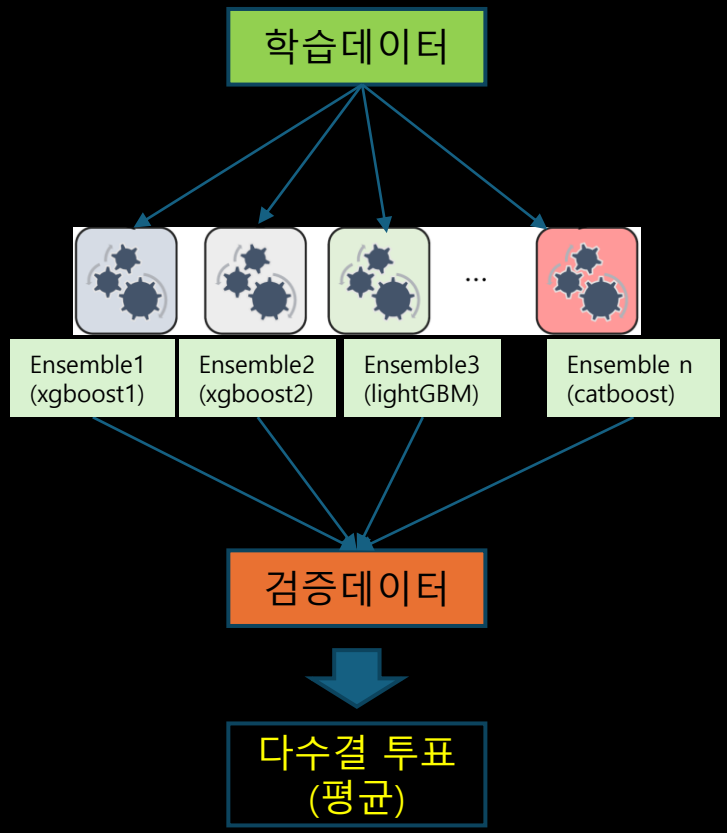
변수1	변수2	SVM	KNN	RF	GBM
0.2	0.1	불량	정상	불량	정상
0.6	-0.3	정상	정상	정상	정상
-0.7	0.2	정상	불량	불량	불량
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:

기존 검증데이터 각 모델별 예측값

Ensemble of Ensembles

Ensemble 모델에 사용되는 learner를 Ensemble Learner로 대치하는 기법





Exercise with Codes

Exercise:

1. Stacking

Thank you!