



# Ensemble

강필성

고려대학교 산업경영공학부

pilsung\_kang@korea.ac.kr

# 양상블: 배경

- No Free Lunch Theorem

- ✓ 어떤 알고리즘도 모든 상황에서 다른 알고리즘보다 우월하다는 결론을 내릴 수 없다.
- ✓ 문제의 목적, 데이터 형태 등을 종합적으로 고려하여 최적의 알고리즘을 선택할 필요가 있음

Journal of Machine Learning Research 15 (2014) 3133-3181

Submitted 11/13; Revised 4/14; Published 10/14

## Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?

**Manuel Fernández-Delgado**

MANUEL.FERNANDEZ.DELGADO@USC.ES

**Eva Cernadas**

EVA.CERNADAS@USC.ES

**Senén Barro**

SEHEN.BARRO@USC.ES

*CITIUS: Centro de Investigación en Tecnoloxías da Información da USC  
University of Santiago de Compostela  
Campus Vida, 15872, Santiago de Compostela, Spain*

**Dinani Amorim**

DINANIAMORIM@GMAIL.COM

*Departamento de Tecnologia e Ciências Sociais- DTCS  
Universidade do Estado da Bahia  
Av. Edgard Chastinet S/N - São Geraldo - Juazeiro-BA, CEP: 48.305-680, Brasil*

# 양상블: 배경

- 집단 지성의 힘!

✓ 단일 알고리즘보다 여러 알고리즘을 결합하면 성능이 향상되는 경우가 많음

Rank	Acc.	$\kappa$	Classifier	Rank	Acc.	$\kappa$	Classifier
32.9	82.0	63.5	parRF_t (RF)	67.3	77.7	55.6	pda_t (DA)
33.1	82.3	63.6	rf_t (RF)	67.6	78.7	55.2	elm_m (NNET)
36.8	81.8	62.2	svm_C (SVM)	67.6	77.8	54.2	SimpleLogistic_w (LMR)
38.0	81.2	60.1	svmPoly_t (SVM)	69.2	78.3	57.4	MAB_J48_w (BST)
39.4	81.9	62.5	rforest_R (RF)	69.8	78.8	56.7	BG_REPTree_w (BAG)
39.6	82.0	62.0	elm_kernel_m (NNET)	69.8	78.1	55.4	SMO_w (SVM)
40.3	81.4	61.1	svmRadialCost_t (SVM)	70.6	78.3	58.0	MLP_w (NNET)
42.5	81.0	60.0	svmRadial_t (SVM)	71.0	78.8	58.23	BG_RandomTree_w (BAG)
42.9	80.6	61.0	C5.0_t (BST)	71.0	77.1	55.1	mlm_R (GLM)
44.1	79.4	60.5	avNNet_t (NNET)	71.0	77.8	56.2	BG_J48_w (BAG)
45.5	79.5	61.0	net_t (NNET)	72.0	75.7	52.6	rbf_t (NNET)
47.0	78.7	59.4	pecaNNet_t (NNET)	72.1	77.1	54.8	fda_R (DA)
47.1	80.8	53.0	BG.LibSVM_w (BAG)	72.4	77.0	54.7	lda_R (DA)
47.3	80.3	62.0	mlp_t (NNET)	72.4	79.1	55.6	svmlight_C (NNET)
47.6	80.6	60.0	RotationForest_w (RF)	72.6	78.4	57.9	AdaBoostM1_J48_w (BST)
50.1	80.9	61.6	RRF_t (RF)	72.7	78.4	56.2	BG_IBk_w (BAG)
51.6	80.7	61.4	RRFglobal_t (RF)	72.9	77.1	54.6	ldaBag_R (BAG)
52.5	80.6	58.0	MAB.LibSVM_w (BST)	73.2	78.3	56.2	BG_LWL_w (BAG)
52.6	79.9	56.9	LibSVM_w (SVM)	73.7	77.9	56.0	MAB.REPTree_w (BST)
57.6	79.1	59.3	adaBoost_R (BST)	74.0	77.4	52.6	RandomSubSpace_w (DT)
58.5	79.7	57.2	pnn_m (NNET)	74.4	76.9	54.2	lda2_t (DA)
58.9	78.5	54.7	cforest_t (RF)	74.6	74.1	51.8	svmBag_R (BAG)
59.9	79.7	42.6	dkp_C (NNET)	74.6	77.5	55.2	LibLINEAR_R_w (SVM)
60.4	80.1	55.8	gaussPrRadialLR (OM)	75.9	77.2	55.6	rbfDDA_t (NNET)
60.5	80.0	57.4	RandomForest_w (RF)	76.5	76.9	53.8	sda_t (DA)
62.1	78.7	56.0	svmLinear_t (SVM)	76.6	78.1	56.5	END_w (OEN)
62.5	78.4	57.5	fda_t (DA)	76.6	77.3	54.8	LogitBoost_w (BST)
62.6	78.6	56.0	knn_t (NN)	76.6	78.2	57.3	MAB.RandomTree_w (BST)
62.8	78.5	58.1	mlp_C (NNET)	77.1	78.4	54.0	BG.RandomForest_w (BAG)
63.0	79.9	59.4	RandomCommittee_w (OEN)	78.5	76.5	53.7	Logistic_w (LMR)
63.4	78.7	58.4	Decorate_w (OEN)	78.7	76.6	50.5	ctreeBag_R (BAG)
63.6	76.9	56.0	mlpWeightDecay_t (NNET)	79.0	76.8	53.5	BG_Logistic_w (BAG)
63.8	78.7	56.7	rda_R (DA)	79.1	77.4	53.0	lvq_t (NNET)
64.0	79.0	58.6	MAB_MLP_w (BST)	79.1	74.4	50.7	pls_t (PLSR)
64.1	79.9	56.9	MAB.RandomForest_w (BST)	79.8	76.9	54.7	hdda_R (DA)
65.0	79.0	56.8	knn_R (NN)	80.6	75.9	53.3	MCC_w (OEN)
65.2	77.9	56.2	multinom_t (LMR)	80.9	76.9	54.5	mda_R (DA)
65.5	77.4	56.6	gevEarth_t (MARS)	81.4	76.7	55.2	C5.0Rules_t (RL)
65.5	77.8	55.7	glmnet_R (GLM)	81.6	78.3	55.8	lssvmRadial_t (SVM)
65.6	78.6	58.4	MAB.PART_w (BST)	81.7	75.6	50.9	JRip_t (RL)
66.0	78.5	56.5	CVR_w (OM)	82.0	76.1	53.3	MAB.Logistic_w (BST)
66.4	79.2	58.9	treebag_t (BAG)	84.2	75.8	53.9	C5.0Tree_t (DT)
66.6	78.2	56.8	BG.PART_w (BAG)	84.6	75.7	50.8	BG.DecisionTable_w (BAG)
66.7	75.5	55.2	mda_t (DA)	84.9	76.5	53.4	NBTree_w (DT)

Rank	Acc.	$\kappa$	Classifier
<b>32.9</b>	82.0	63.5	parRF_t (RF)
33.1	<b>82.3</b>	<b>63.6</b>	rf_t (RF)
36.8	81.8	62.2	svm_C (SVM)
38.0	81.2	60.1	svmPoly_t (SVM)
39.4	81.9	62.5	rforest_R (RF)
39.6	82.0	62.0	elm_kernel_m (NNET)
40.3	81.4	61.1	svmRadialCost_t (SVM)
42.5	81.0	60.0	svmRadial_t (SVM)
42.9	80.6	61.0	C5.0_t (BST)
44.1	79.4	60.5	avNNet_t (NNET)
45.5	79.5	61.0	net_t (NNET)
47.0	78.7	59.4	pecaNNet_t (NNET)
47.1	80.8	53.0	BG.LibSVM_w (BAG)
47.3	80.3	62.0	mlp_t (NNET)
47.6	80.6	60.0	RotationForest_w (RF)
50.1	80.9	61.6	RRF_t (RF)
51.6	80.7	61.4	RRFglobal_t (RF)
52.5	80.6	58.0	MAB.LibSVM_w (BST)
52.6	79.9	56.9	LibSVM_w (SVM)
57.6	79.1	59.3	adaBoost_R (BST)
58.5	79.7	57.2	pnn_m (NNET)
58.9	78.5	54.7	cforest_t (RF)
59.9	79.7	42.6	dkp_C (NNET)
60.4	80.1	55.8	gaussPrRadialLR (OM)
60.5	80.0	57.4	RandomForest_w (RF)
62.1	78.7	56.0	svmLinear_t (SVM)
62.5	78.4	57.5	fda_t (DA)
62.6	78.6	56.0	knn_t (NN)
62.8	78.5	58.1	mlp_C (NNET)
63.0	79.9	59.4	RandomCommittee_w (OEN)
63.4	78.7	58.4	Decorate_w (OEN)
63.6	76.9	56.0	mlpWeightDecay_t (NNET)
63.8	78.7	56.7	rda_R (DA)
64.0	79.0	58.6	MAB_MLP_w (BST)
64.1	79.9	56.9	MAB.RandomForest_w (BST)
65.0	79.0	56.8	knn_R (NN)
65.2	77.9	56.2	multinom_t (LMR)
65.5	77.4	56.6	gevEarth_t (MARS)
65.5	77.8	55.7	glmnet_R (GLM)
65.6	78.6	58.4	MAB.PART_w (BST)
66.0	78.5	56.5	CVR_w (OM)
66.4	79.2	58.9	treebag_t (BAG)
66.6	78.2	56.8	BG.PART_w (BAG)
66.7	75.5	55.2	mda_t (DA)

## 논문의 결론

121개의 공개 데이터셋에 대한 실험 결과

Random Forests (의사결정나무의

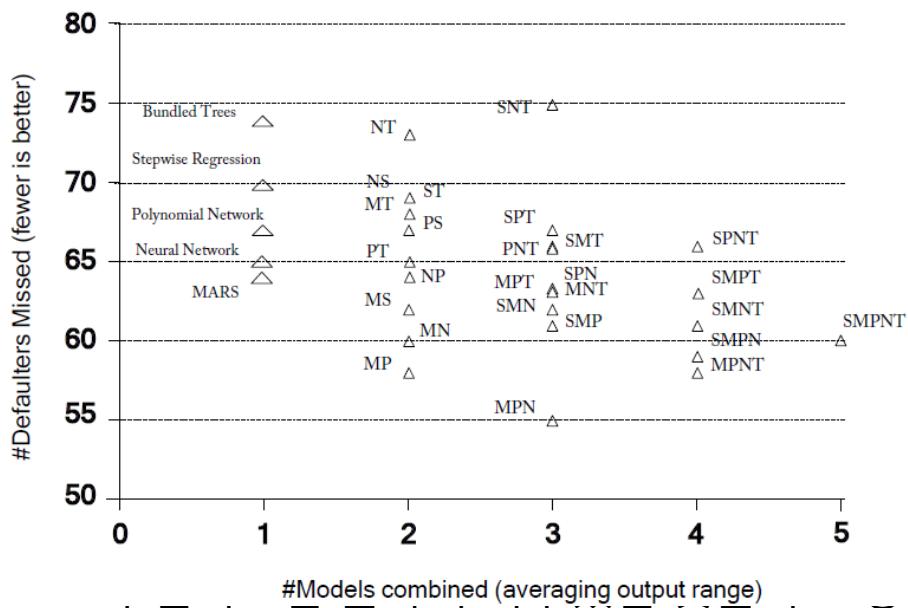
양상블)과 SVM 계열이 상대적으로 분류

성능이 높게 나타남

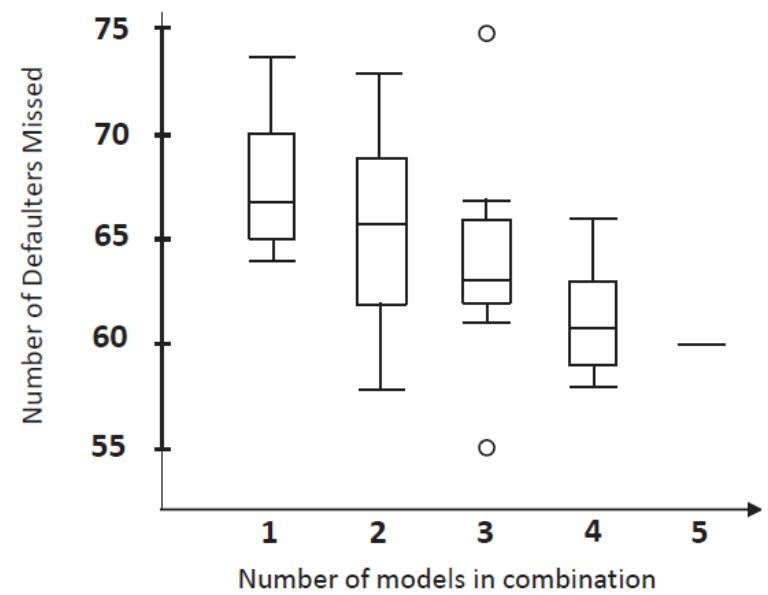
# 양상블: 배경

- 집단 지성의 힘!

- ✓ 단일 알고리즈다 보다 여러 알고리즘을 결합하면 성능이 향상되는 경우가 많음
- ✓ 아래 신용카드 연체 예측에서도 알고리즘을 다양하게 결합할수록 오분류율이 감소하는 현상을 나타냄



방법론을 차용하여 결합



# 양상블: 배경

- The 10 main takeaways from MLConf SF (2016)

- ✓ It's (still) not all about Deep Learning
- ✓ Choose the right problem to solve, with the right metric
- ✓ Fine tuning your models is 5% of a project

**✓ Ensembles almost always work better**

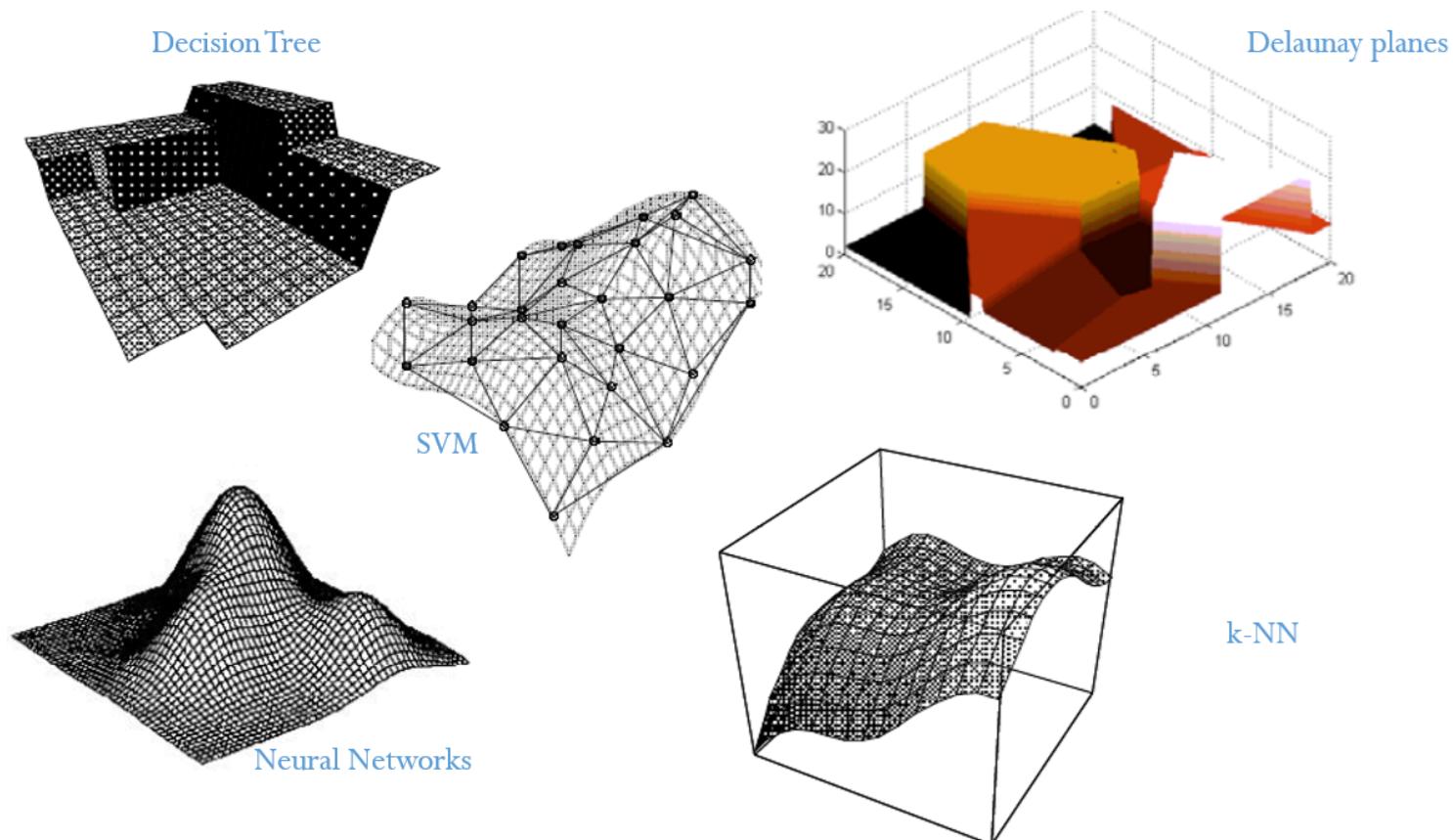
- ✓ The trend towards personalization
- ✓ Manual curation of content is still used in practice
- ✓ Avoid the curse of complexity
- ✓ Learn the best practices from established players
- ✓ Everybody is using open source
- ✓ Make sure you have support from the executives



# 양상블: 배경

- 왜 다양한 알고리즘을 결합하는 것이 효과적인가?

- ✓ 서로 다른 사고방식 체계를 가지고 있어 상호 보완이 가능함
- ✓ 머신 러닝 측면에서는 서로 다른 모델은 서로 다른 분류 경계면/예측 곡선을 생성함



# 양상블: Bias-Variance Decomposition

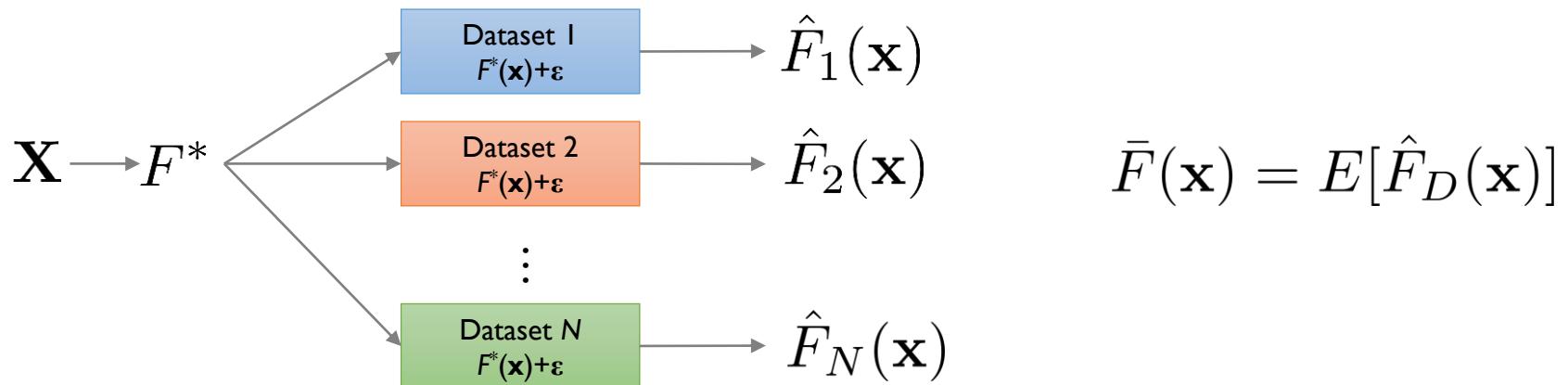
- 실제 데이터는 항상 노이즈가 존재

$$y = F^*(\mathbf{x}) + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$$

- ✓  $F^*(\mathbf{x})$  는 우리가 학습하고 싶은 데이터 생성 함수, 노이즈때문에 정확한 추정은 현실적으로 불가능
- ✓ 노이즈는 서로 독립적이고 일정한 분산을 갖는다고 가정

- 모델 학습

- ✓ 주어진 하나의 샘플 집합으로부터 데이터 생성 함수를 추론하는 것



# 양상블: Bias-Variance Decomposition

- 모델에 의한 오류는 편향(Bias)과 분산(Variance)로 구분될 수 있음
  - ✓ 편향(Bias): 반복적인 모델 학습 시 평균적으로 얼마나 정확한 추정이 가능한지에 대한 측정 지표
  - ✓ 분산(Variance): 반복적인 모델 학습 시 개별 추정이 얼마나 차이가 크게 나타나는지에 대한 측정 지표

$$\begin{aligned} Err(\mathbf{x}_0) &= E \left[ y - \hat{F}(\mathbf{x}) \mid \mathbf{x} = \mathbf{x}_0 \right]^2 \\ &= \left[ \hat{F}^*(\mathbf{x}_0) - \bar{F}(\mathbf{x}_0) \right]^2 + E \left[ \bar{F}(\mathbf{x}_0) - \hat{F}(\mathbf{x}_0) \right]^2 + \sigma^2 \\ &= Bias^2(\hat{F}(\mathbf{x}_0)) + Var(\hat{F}(\mathbf{x}_0)) + \sigma^2 \end{aligned}$$

# Bias-Variance Decomposition 유도(optional)

- The MSE for a particular data point

$$\begin{aligned} Err(\mathbf{x}_0) &= E \left[ y - \hat{F}(\mathbf{x}) \mid \mathbf{x} = \mathbf{x}_0 \right]^2 && (y = F^*(\mathbf{x}) + \epsilon) \\ &= E \left[ \hat{F}^*(\mathbf{x}_0) + \epsilon - \hat{F}(\mathbf{x}_0) \right]^2 \\ &= E \left[ \hat{F}^*(\mathbf{x}_0) - \hat{F}(\mathbf{x}_0) \right]^2 + \sigma^2 \\ &= E \left[ \hat{F}^*(\mathbf{x}_0) - \bar{F}(\mathbf{x}_0) + \bar{F}(\mathbf{x}_0) - \hat{F}(\mathbf{x}_0) \right]^2 + \sigma^2 \end{aligned}$$

# Bias-Variance Decomposition 유도(optional)

- The MSE for a particular data point

$$= E \left[ \hat{F}^*(\mathbf{x}_0) - \bar{F}(\mathbf{x}_0) + \bar{F}(\mathbf{x}_0) - \hat{F}(\mathbf{x}_0) \right]^2 + \sigma^2$$

✓ By the properties of the expectation operator

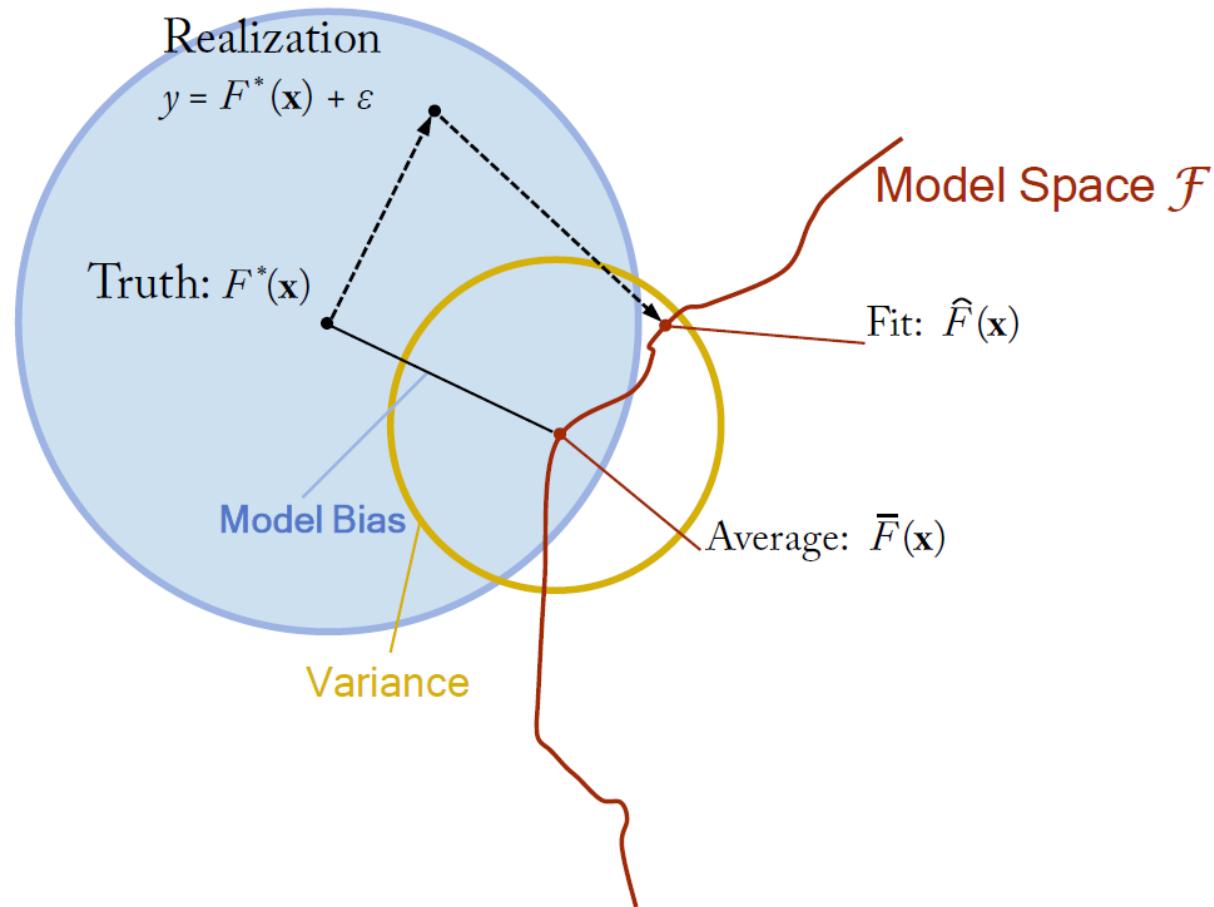
$$= E \left[ \hat{F}^*(\mathbf{x}_0) - \bar{F}(\mathbf{x}_0) \right]^2 + E \left[ \bar{F}(\mathbf{x}_0) - \hat{F}(\mathbf{x}_0) \right]^2 + \sigma^2$$

$$= \left[ \hat{F}^*(\mathbf{x}_0) - \bar{F}(\mathbf{x}_0) \right]^2 + E \left[ \bar{F}(\mathbf{x}_0) - \hat{F}(\mathbf{x}_0) \right]^2 + \sigma^2$$

$$= Bias^2(\hat{F}(\mathbf{x}_0)) + Var(\hat{F}(\mathbf{x}_0)) + \sigma^2$$

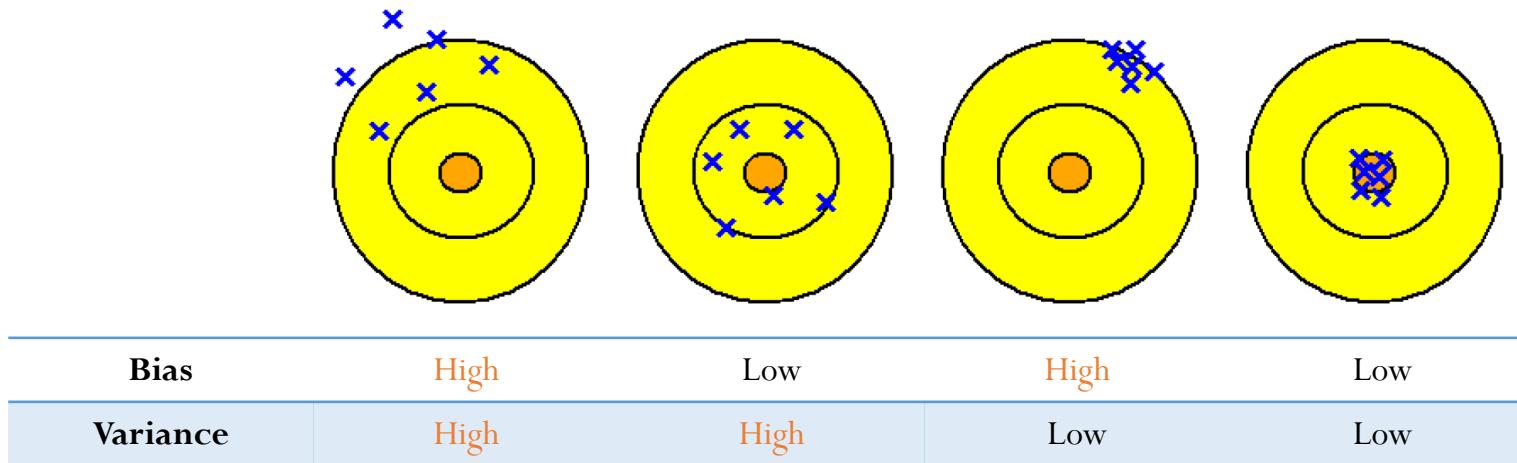
# 양상블: Bias-Variance Decomposition

- 편향(Bias)과 분산(Variance)의 기하학적 측면

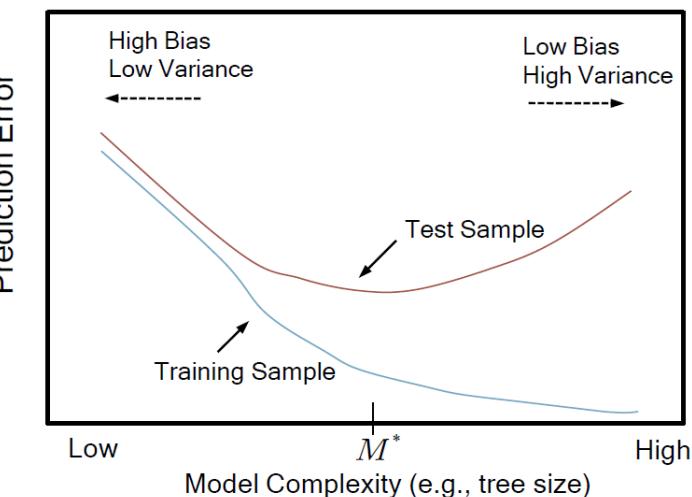


# 양상블: Bias-Variance Decomposition

- 편향(Bias)과 분산(Variance)의 크기에 따른 모델의 구분



- ✓ 낮은 모델 복잡도: 높은 편향 & 낮은 분산
  - Logistic regression, LDA, k-NN with large k, etc.
- ✓ 높은 모델 복잡도: 낮은 편향 & 높은 분산
  - DT, ANN, SVM, k-NN with small k, etc.

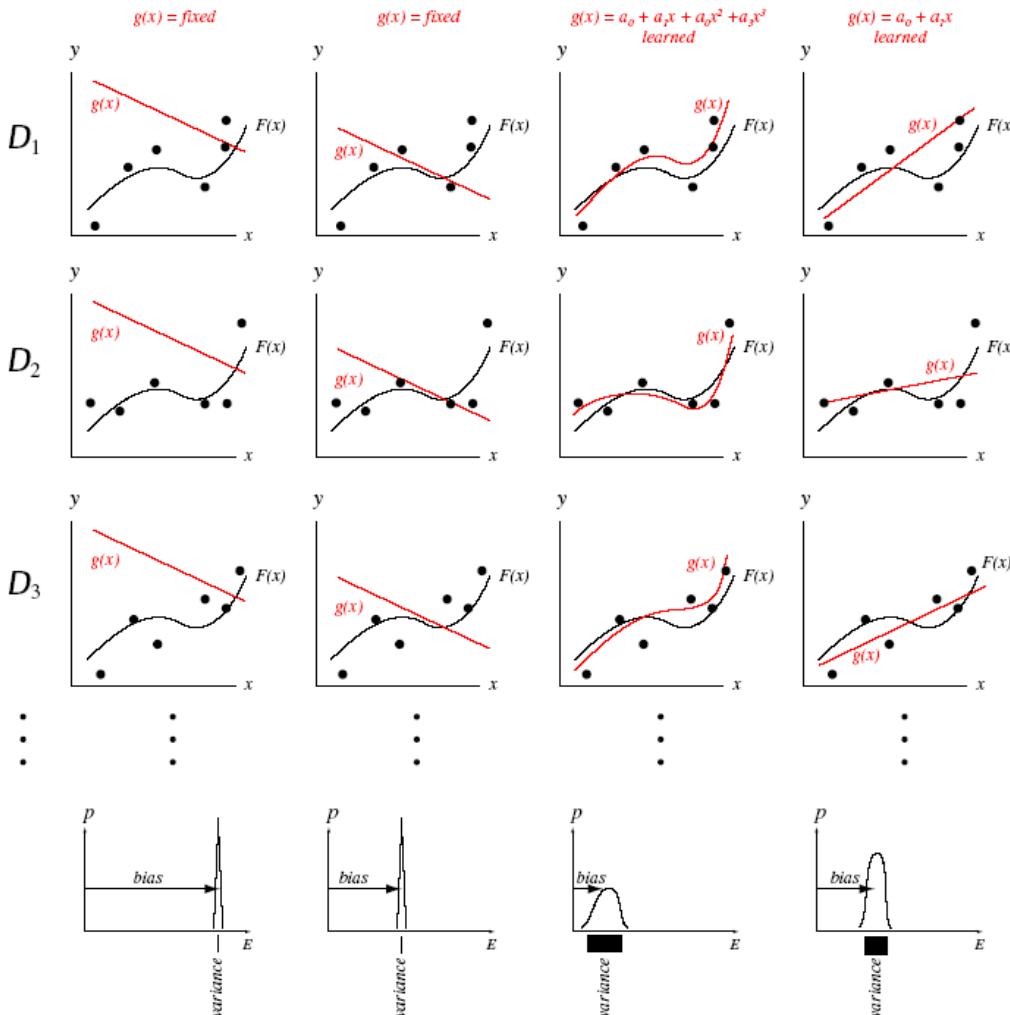


# 양상분: Bias-Variance Decomposition

- 편향(Bias)과 분산(Variance)에 따른 추정 함수 예시

Each column is a different model.

Each row is a different dataset of 6 points.



Histograms of mean-squared error of the fit.

Col 1:

Poor fixed linear model;  
High bias, zero variance

Col 2:

Slightly better fixed linear model;  
Lower (but high) bias, zero variance.

Col 3:

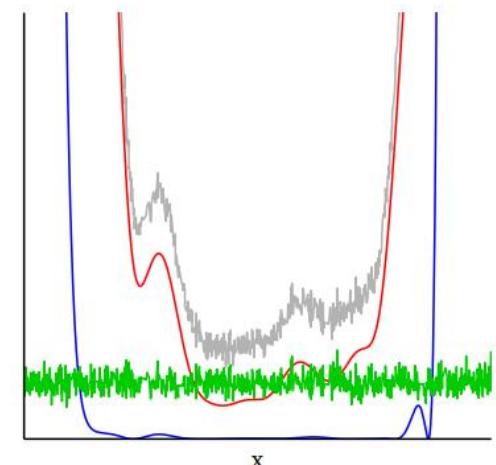
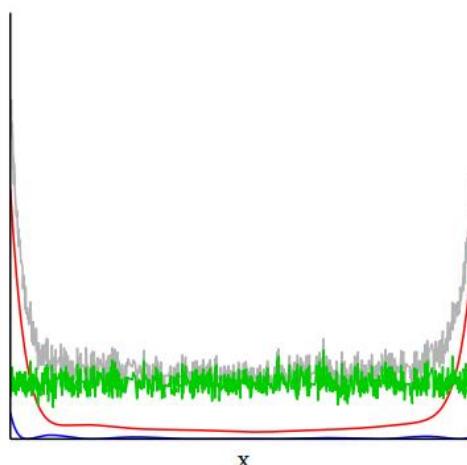
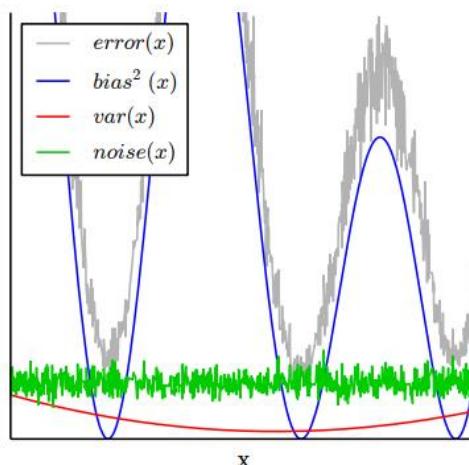
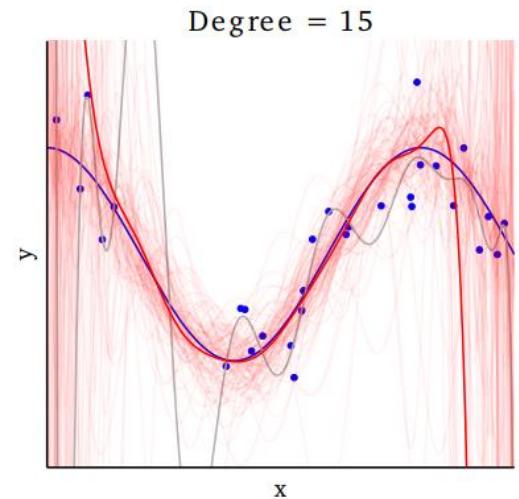
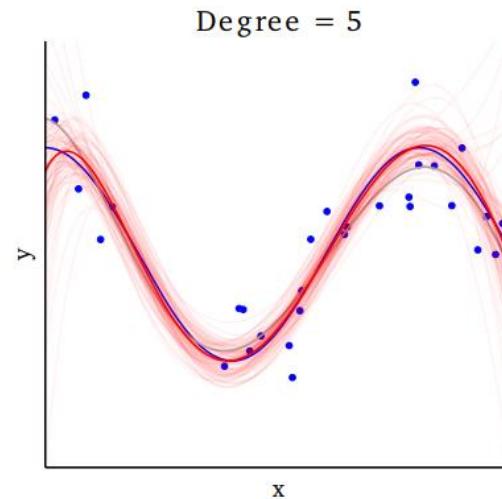
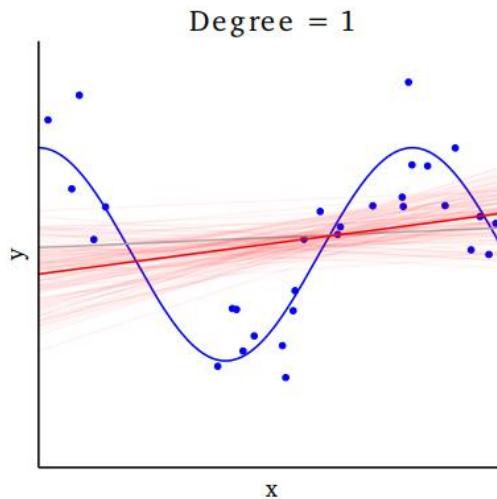
Learned cubic model;  
Low bias, moderate variance.

Col 4:

Learned linear model;  
Intermediate bias and variance.

# 양상블: Bias-Variance Decomposition

- 편향(Bias)과 분산(Variance)에 따른 추정 함수 예시



# 앙상블: 목적

- 앙상블의 목적: 다수의 모델을 학습하여 오류의 감소를 추구
  - ✓ 분산의 감소에 의한 오류 감소: 배깅(Bagging), 랜덤 포레스트(Random Forest)
  - ✓ 편향의 감소에 의한 오류 감소: 부스팅(Boosting)
  - ✓ 분산과 편향의 동시 감소: Mixture of Experts
- 앙상블 구성의 두 가지 핵심 아이디어
  - ✓ 다양성(diversity)을 어떻게 확보할 것인가?
  - ✓ 최종 결과물을 어떻게 결합(combine, aggregate)할 것인가?

# 양상블: 효과

- 양상블의 효과

- ✓ 이론적으로는  $M$ 개의 개별 모델을 결합한 양상블의 경우  $M$ 개의 개별 모델의 평균 오류의  $1/M$  수준으로 오류가 감소함 (가정: 각 모델은 서로 독립)

$$E_{Ensemble} = \frac{1}{M} E_{Avg}$$

- ✓ 위 가정은 현실세계에서 지켜지지 않는 경우가 많음
- ✓ 현실적으로는  $M$ 개의 개별 모델을 결합한 양상블의 경우 개별 모델의 평균 오류보다는 최소한 같거나 낮은 오류를 나타내는 것을 증명할 수 있음

$$\left[ \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right]^2 \leq M \sum_{m=1}^M \epsilon_m(\mathbf{x})^2 \Rightarrow \left[ \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x}) \right]^2 \leq \frac{1}{M} \sum_{m=1}^M \epsilon_m(\mathbf{x})^2$$

$$E_{Ensemble} \leq E_{Avg}$$

- 즉, 1등 모델보다 성능이 우수함을 입증할 수는 없으나 개별 모델들의 평균치보다는 항상 우수하거나 같은 성능을 나타냄

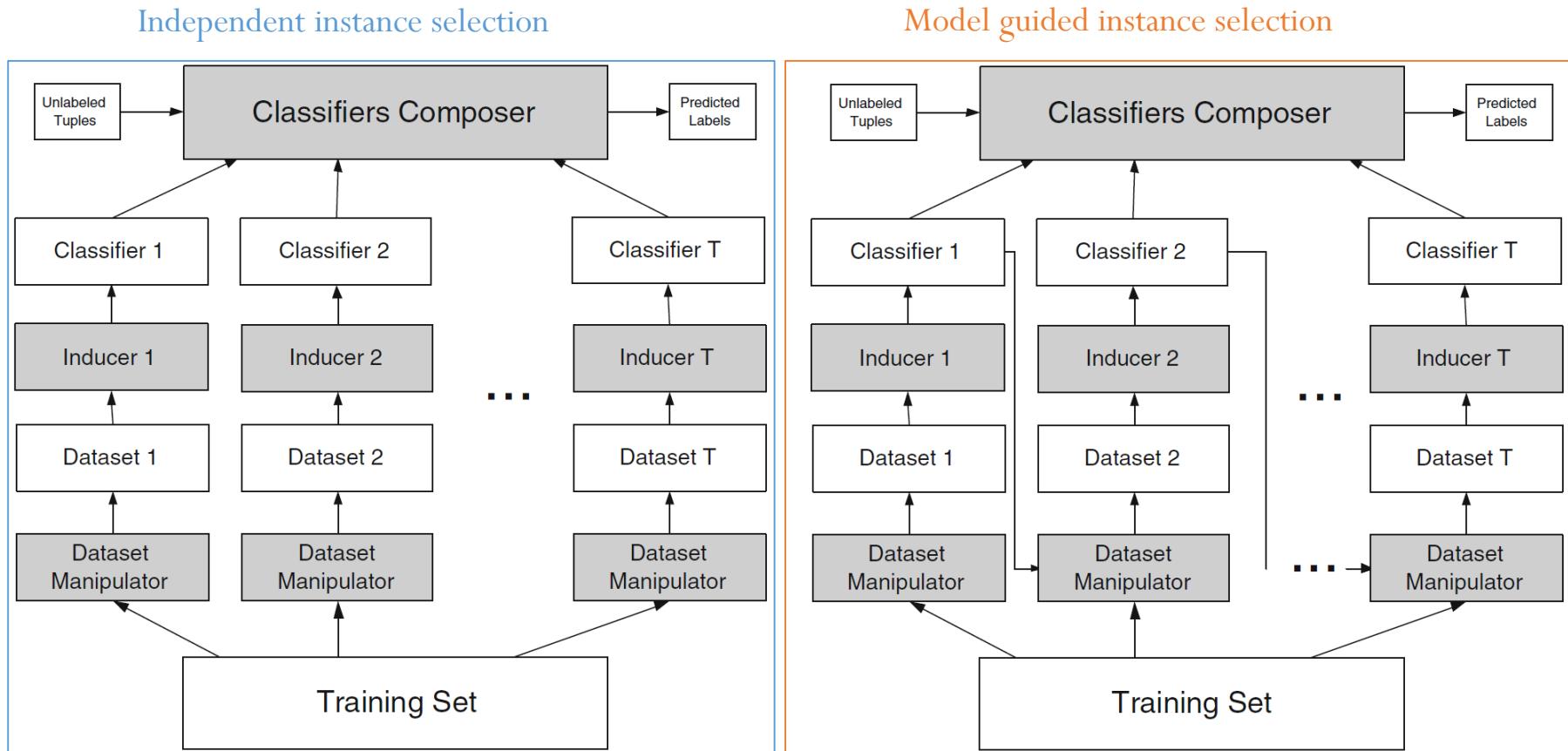
# 양상블의 다양성(Diversity)

- 동일한 모델을 여러 개 사용하는 것은 아무런 효과가 없음
  - ✓ 개별 모델은 서로 적절하게 달라야 양상블의 효과를 볼 수 있음
  - ✓ 개별적으로는 어느 정도 좋은 성능을 가지면서, 양상블 내에서 각각의 모델은 서로 다양한 형태를 나타내는 것이 가장 이상적임

Diversity	Implicit	Explicit
Description	Provide different random subset of the training data to each learner	Use some measurement ensuring it is substantially different from the other members
Ensemble Algorithms	Instance: Bagging Variables: Random Subspaces, Rotation Forests Both: Random Forests	Boosting, Negative Correlation Learning

# 양상블의 다양성

- Independent (implicit) vs. Model guided (explicit) instance selection



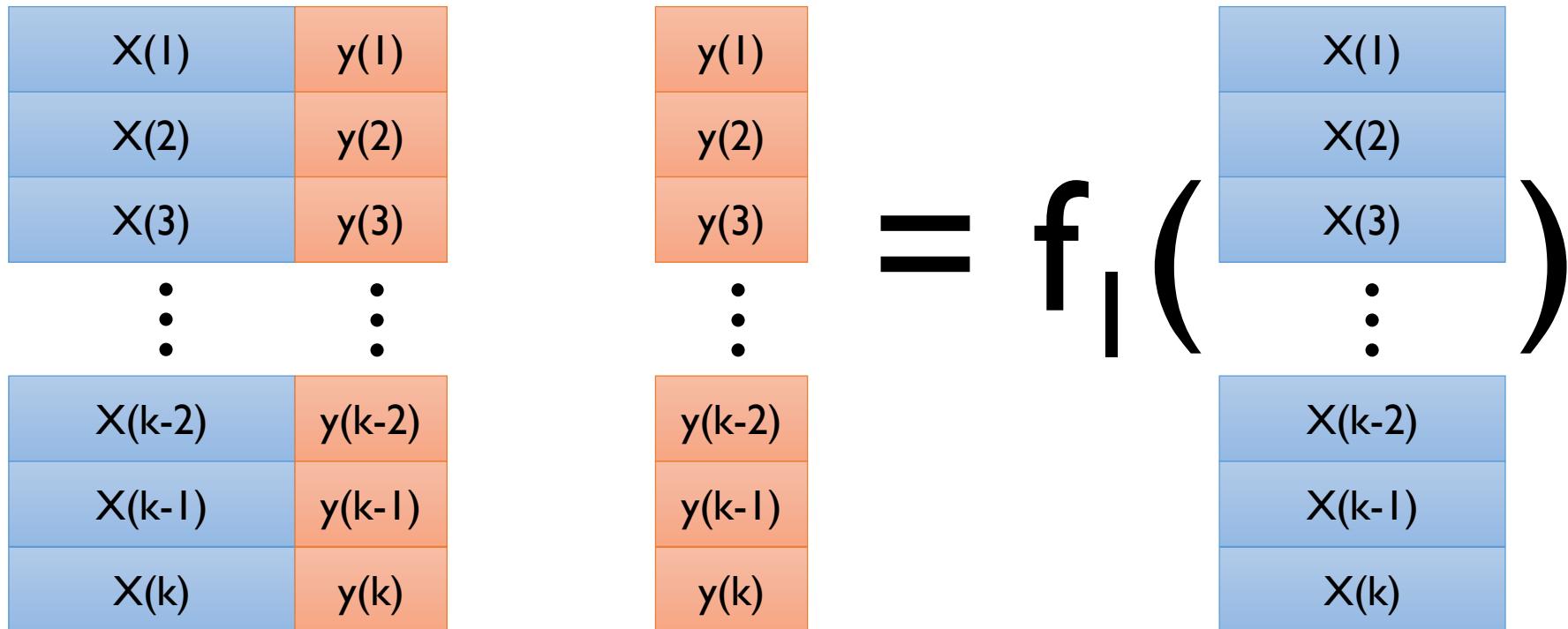
# AGENDA

- 01 양상블: 배경
- 02 배깅: Bagging & Random Forests
- 03 부스팅: AdaBoost & GBM

# 비복원 추출을 통한 양상을

- K-fold data split

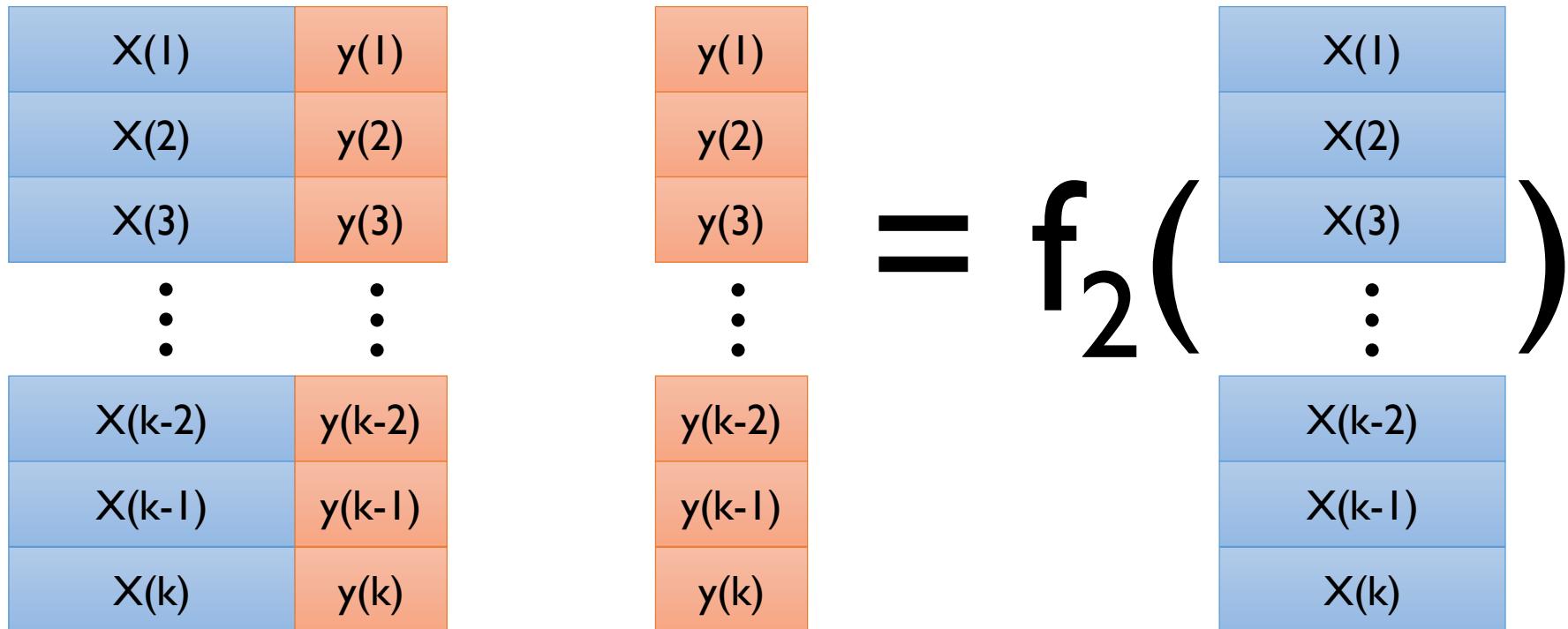
- ✓ 전체 데이터를 k개의 block으로 나누고 개별 모델을 서로 다른 (k-1)개의 subset에 대해 학습한 뒤 최종 결과를 결합



# 비복원 추출을 통한 양상을

- K-fold data split

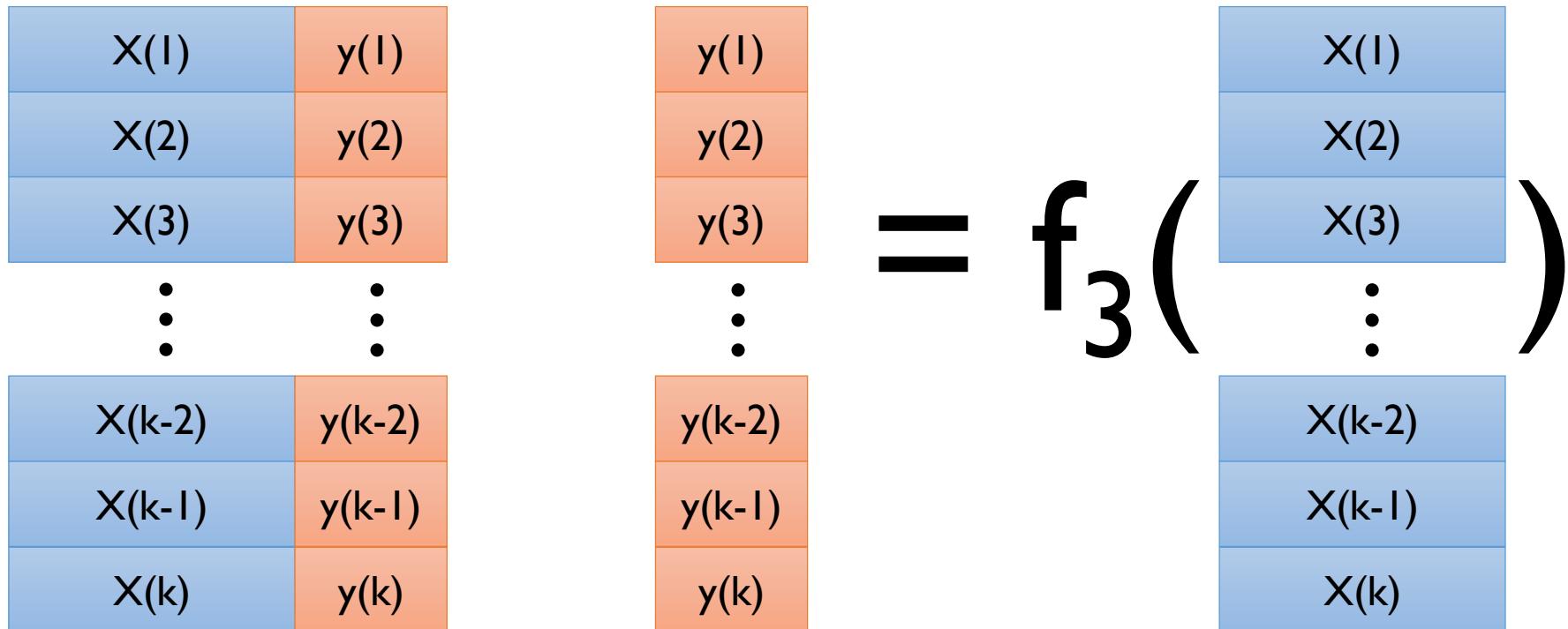
- ✓ 전체 데이터를 k개의 block으로 나누고 개별 모델을 서로 다른 (k-1)개의 subset에 대해 학습한 뒤 최종 결과를 결합



# 비복원 추출을 통한 양상을

- K-fold data split

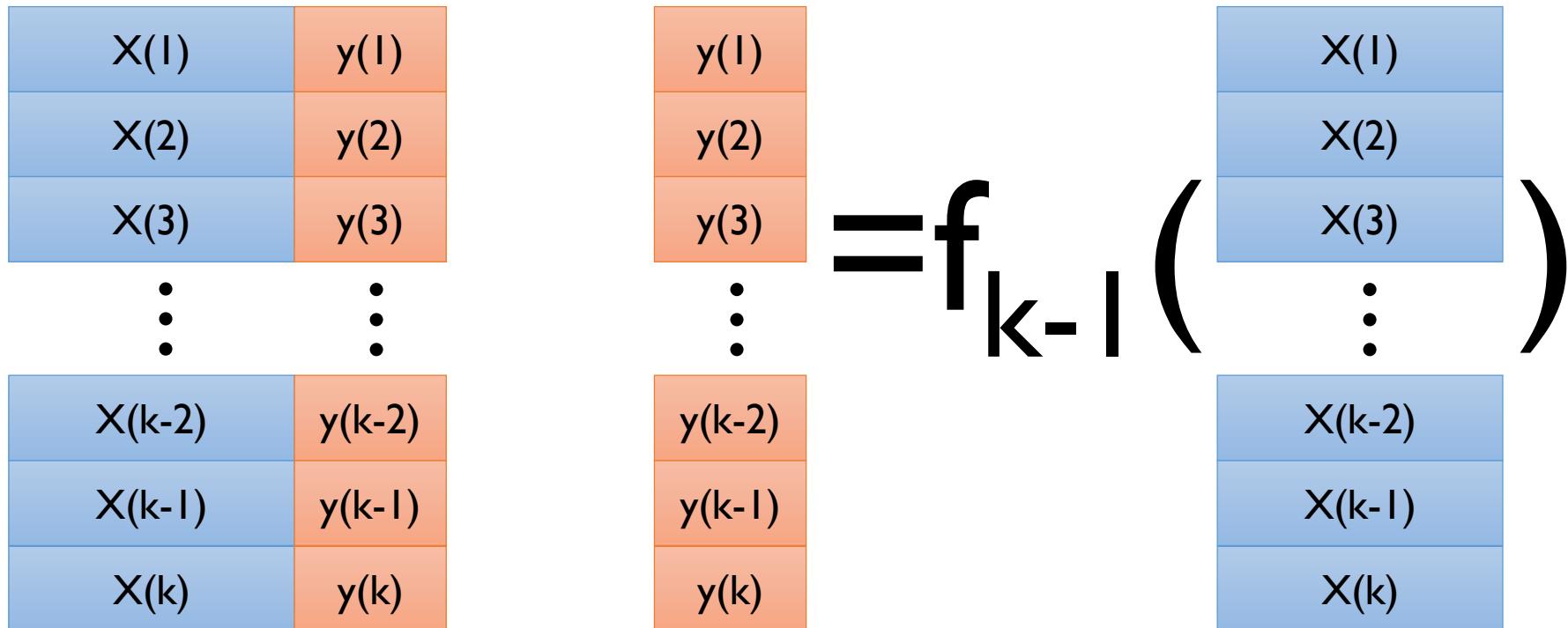
- ✓ 전체 데이터를 k개의 block으로 나누고 개별 모델을 서로 다른 (k-1)개의 subset에 대해 학습한 뒤 최종 결과를 결합



# 비복원 추출을 통한 양상을

- K-fold data split

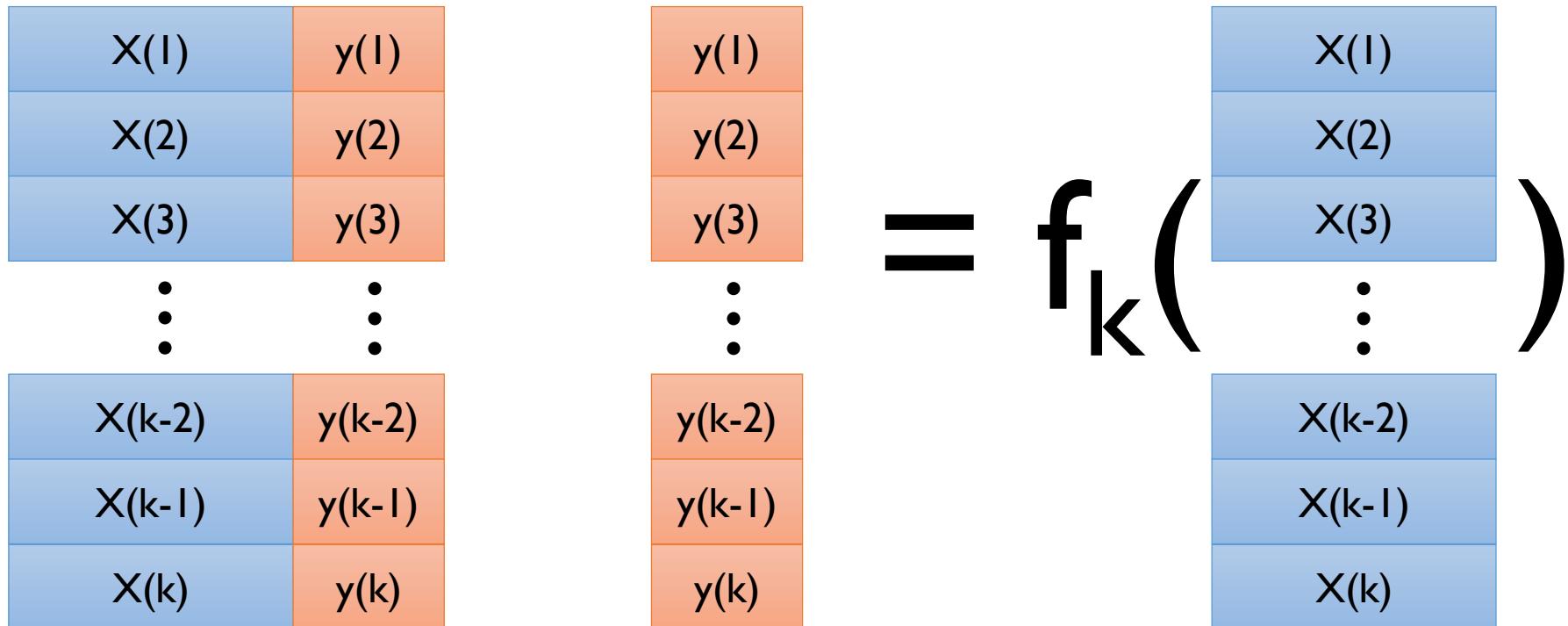
- ✓ 전체 데이터를 k개의 block으로 나누고 개별 모델을 서로 다른 (k-1)개의 subset에 대해 학습한 뒤 최종 결과를 결합



# 비복원 추출을 통한 양상을

- K-fold data split

- ✓ 전체 데이터를 k개의 block으로 나누고 개별 모델을 서로 다른 (k-1)개의 subset에 대해 학습한 뒤 최종 결과를 결합



# 비복원 추출을 통한 양상을

- K-fold data split

- ✓ 전체 데이터를 k개의 block으로 나누고 개별 모델을 서로 다른 (k-1)개의 subset에 대해 학습한 뒤 최종 결과를 결합
- ✓ 최종 예측 값

$$\hat{y} = \delta(f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_{k-1}(\mathbf{x}), f_k(\mathbf{x}))$$

- $\delta(\cdot)$  : 개별 모델의 결과를 취합하는 함수(aggregation function)

# 배깅: Bagging

- Bagging: Bootstrapping Aggregating

- ✓ 양상블의 각 멤버(모델)은 서로 다른 학습 데이터셋을 이용
- ✓ 각 데이터셋은 복원 추출(sampling with replacement)을 통해 원래 데이터의 수만큼의 크기를 갖도록 샘플링
- ✓ 개별 데이터셋을 브스트랩(bootstrap)이라 부름

Original Dataset

$x^1$	$y^1$
$x^2$	$y^2$
$x^3$	$y^3$
$x^4$	$y^4$
$x^5$	$y^5$
$x^6$	$y^6$
$x^7$	$y^7$
$x^8$	$y^8$
$x^9$	$y^9$
$x^{10}$	$y^{10}$

Bootstrap 1

$x^3$	$y^3$
$x^6$	$y^6$
$x^2$	$y^2$
$x^{10}$	$y^{10}$
$x^8$	$y^8$
$x^7$	$y^7$
$x^7$	$y^7$
$x^3$	$y^3$
$x^2$	$y^2$
$x^7$	$y^7$

Bootstrap 2

$x^7$	$y^7$
$x^1$	$y^1$
$x^{10}$	$y^{10}$
$x^1$	$y^1$
$x^8$	$y^8$
$x^6$	$y^6$
$x^2$	$y^2$
$x^6$	$y^6$
$x^4$	$y^4$
$x^9$	$y^9$

...

Bootstrap B

$x^9$	$y^9$
$x^5$	$y^5$
$x^2$	$y^2$
$x^4$	$y^4$
$x^7$	$y^7$
$x^2$	$y^2$
$x^5$	$y^5$
$x^{10}$	$y^{10}$
$x^8$	$y^8$
$x^2$	$y^2$

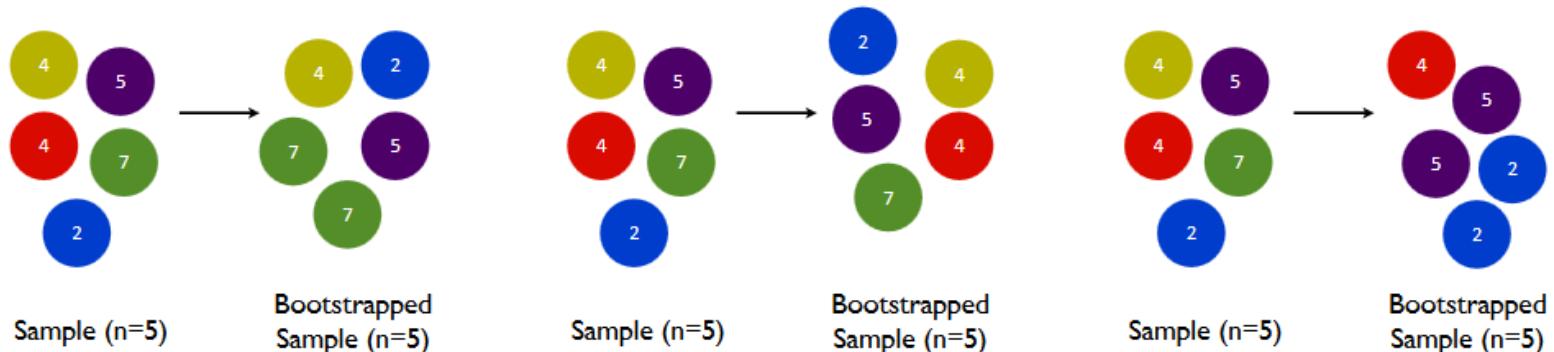
# 배깅: Bagging

- Bagging: Bootstrapp Aggregating

- ✓ 이론적으로 한 개체가 하나의 븋스트랩에 한번도 선택되지 않을 확률

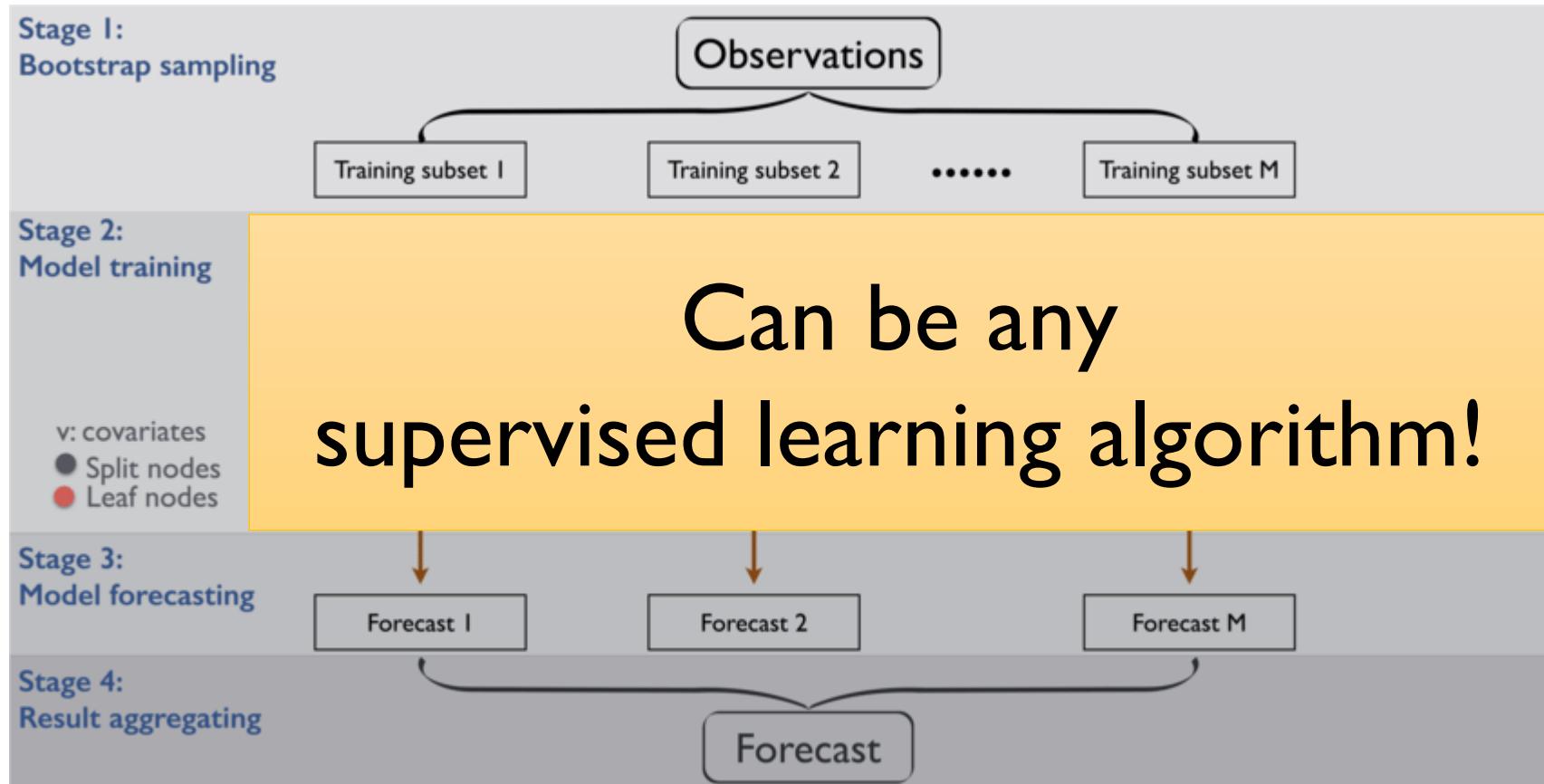
$$p = \left(1 - \frac{1}{N}\right)^N \rightarrow \lim_{N \rightarrow \infty} \left(1 - \frac{1}{N}\right)^N = e^{-1} = 0.368$$

- ✓ 개별 모델의 분산은 높고 편향이 낮은 알고리즘에 적절함 (인공신경망 등)



# 배깅: Bagging

- Bagging with Decision Tree



# 배깅: Bagging

- Result Aggregating
  - ✓ For classification problem
    - Majority voting

$$\hat{y}_{Ensemble} = \arg \max_i \left( \sum_{j=1}^n \delta(\hat{y}_j = i), \quad i \in \{0, 1\} \right)$$

Training Accuracy	Ensemble population	$P(y=1)$ for a test instance	Predicted class label
0.80	Model 1	0.90	1
0.75	Model 2	0.92	1
0.88	Model 3	0.87	1
0.91	Model 4	0.34	0
0.77	Model 5	0.41	0
0.65	Model 6	0.84	1
0.95	Model 7	0.14	0
0.82	Model 8	0.32	0
0.78	Model 9	0.98	1
0.83	Model 10	0.57	1

$$\sum_{j=1}^n \delta(\hat{y}_j = 0) = 4$$

$$\sum_{j=1}^n \delta(\hat{y}_j = 1) = 6$$

$$\hat{y}_{Ensemble} = 1$$

# 배깅: Bagging

- Result Aggregating

- ✓ For classification problem

- Weighted voting (weight = training accuracy of individual models)

$$\hat{y}_{Ensemble} = \arg \max_i \left( \frac{\sum_{j=1}^n (TrnAcc_j) \cdot \delta(\hat{y}_j = i)}{\sum_{j=1}^n (TrnAcc_j)}, \quad i \in \{0, 1\} \right)$$

Training Accuracy	Ensemble population	P(y=1) for a test instance	Predicted class label
0.80	Model 1	0.90	1
0.75	Model 2	0.92	1
0.88	Model 3	0.87	1
0.91	Model 4	0.34	0
0.77	Model 5	0.41	0
0.65	Model 6	0.84	1
0.95	Model 7	0.14	0
0.82	Model 8	0.32	0
0.78	Model 9	0.98	1
0.83	Model 10	0.57	1

$$\frac{\sum_{j=1}^n (TrnAcc_j) \cdot \delta(\hat{y}_j = 0)}{\sum_{j=1}^n (TrnAcc_j)} = 0.424$$

$$\frac{\sum_{j=1}^n (TrnAcc_j) \cdot \delta(\hat{y}_j = 1)}{\sum_{j=1}^n (TrnAcc_j)} = 0.576$$

$$\hat{y}_{Ensemble} = 1$$

# 배깅: Bagging

- Result Aggregating

- ✓ For classification problem

- Weighted voting (weight = predicted probability for each class)

$$\hat{y}_{Ensemble} = \arg \max_i \left( \frac{1}{n} \sum_{j=1}^n P(y = i), \quad i \in \{0, 1\} \right)$$

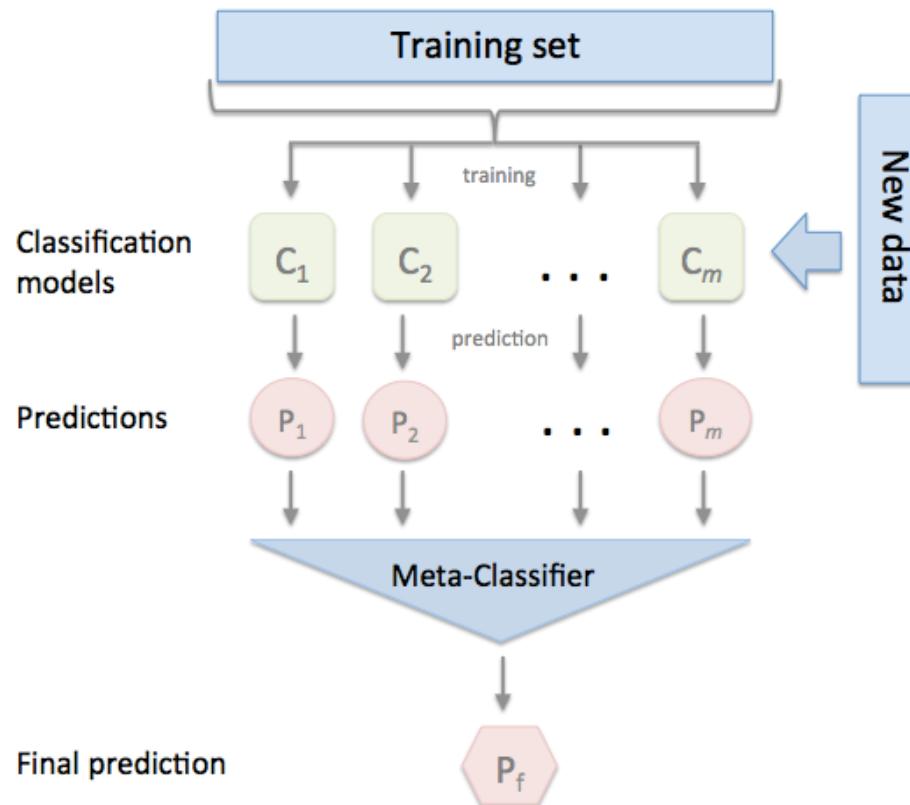
Training Accuracy	Ensemble population	P(y=1) for a test instance	Predicted class label
0.80	Model 1	0.90	1
0.75	Model 2	0.92	1
0.88	Model 3	0.87	1
0.91	Model 4	0.34	0
0.77	Model 5	0.41	0
0.65	Model 6	0.84	1
0.95	Model 7	0.14	0
0.82	Model 8	0.32	0
0.78	Model 9	0.98	1
0.83	Model 10	0.57	1

$\sum_{j=1}^n P(y = 0) = 0.375$   
 $\sum_{j=1}^n P(y = 1) = 0.625$

$\hat{y}_{Ensemble} = 1$

# 배깅: Bagging

- Result Aggregating: Stacking
  - ✓ Use another prediction model to aggregate the results
    - Input: Predictions made by ensemble members
    - Target: Actual true label



# 배깅: Bagging

- Result Aggregating: Stacking
  - ✓ The winner of KDD-cup 2015
    - MOOC dropout prediction



• Jeong-Yoon Lee, Winning Data Science Competitions

# 배깅: Bagging

- Bagging: Algorithm

---

## Algorithm 1 Bagging

---

**Input:** Required ensemble size  $T$

**Input:** Training set  $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$

**for**  $t = 1$  to  $T$  **do**

    Build a dataset  $S_t$ , by sampling  $N$  items, randomly *with replacement* from  $S$ .

    Train a model  $h_t$  using  $S_t$ , and add it to the ensemble.

**end for**

For a new testing point  $(x', y')$ ,

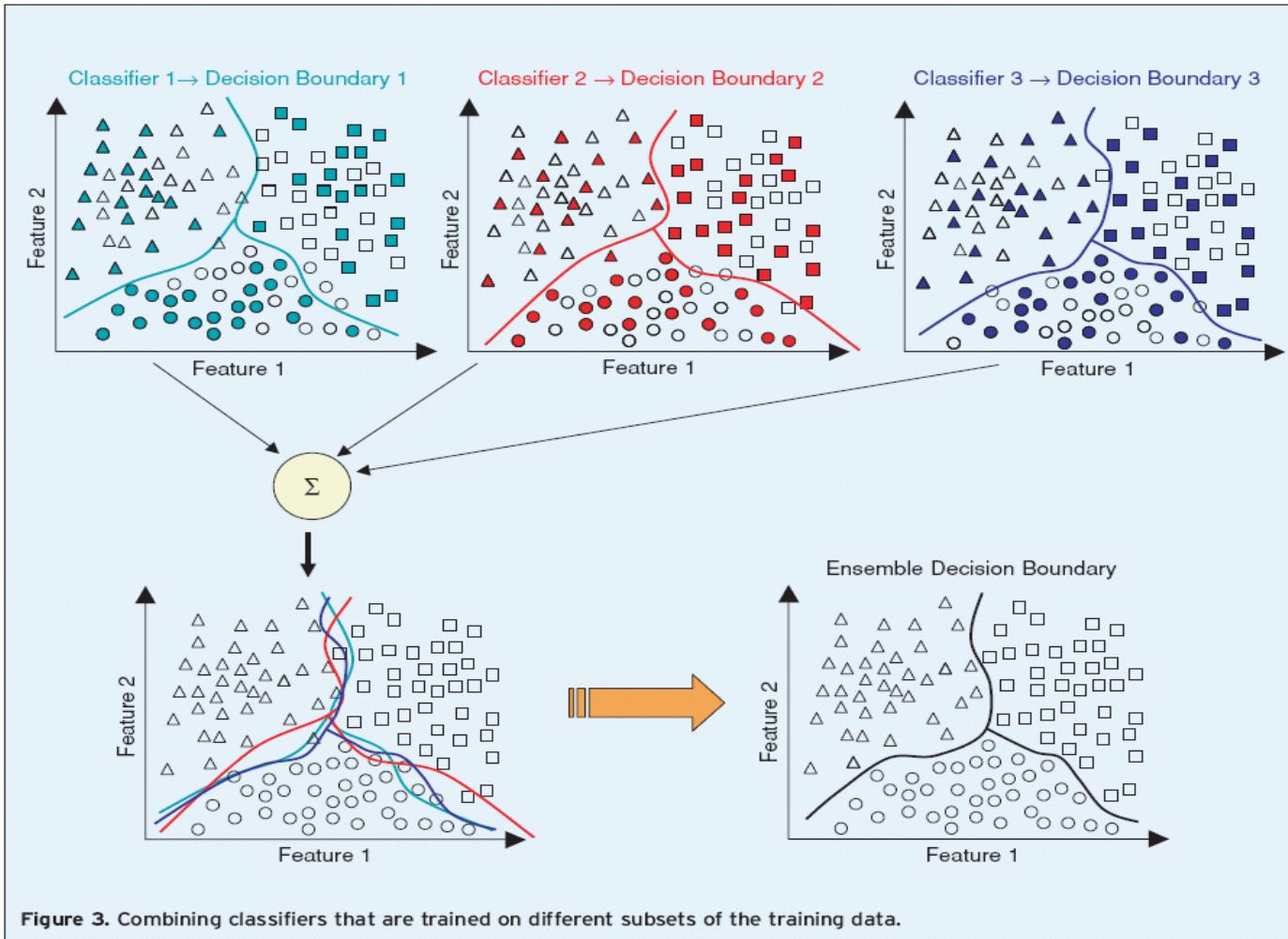
If model outputs are continuous, combine them by averaging.

If model outputs are class labels, combine them by voting.

---

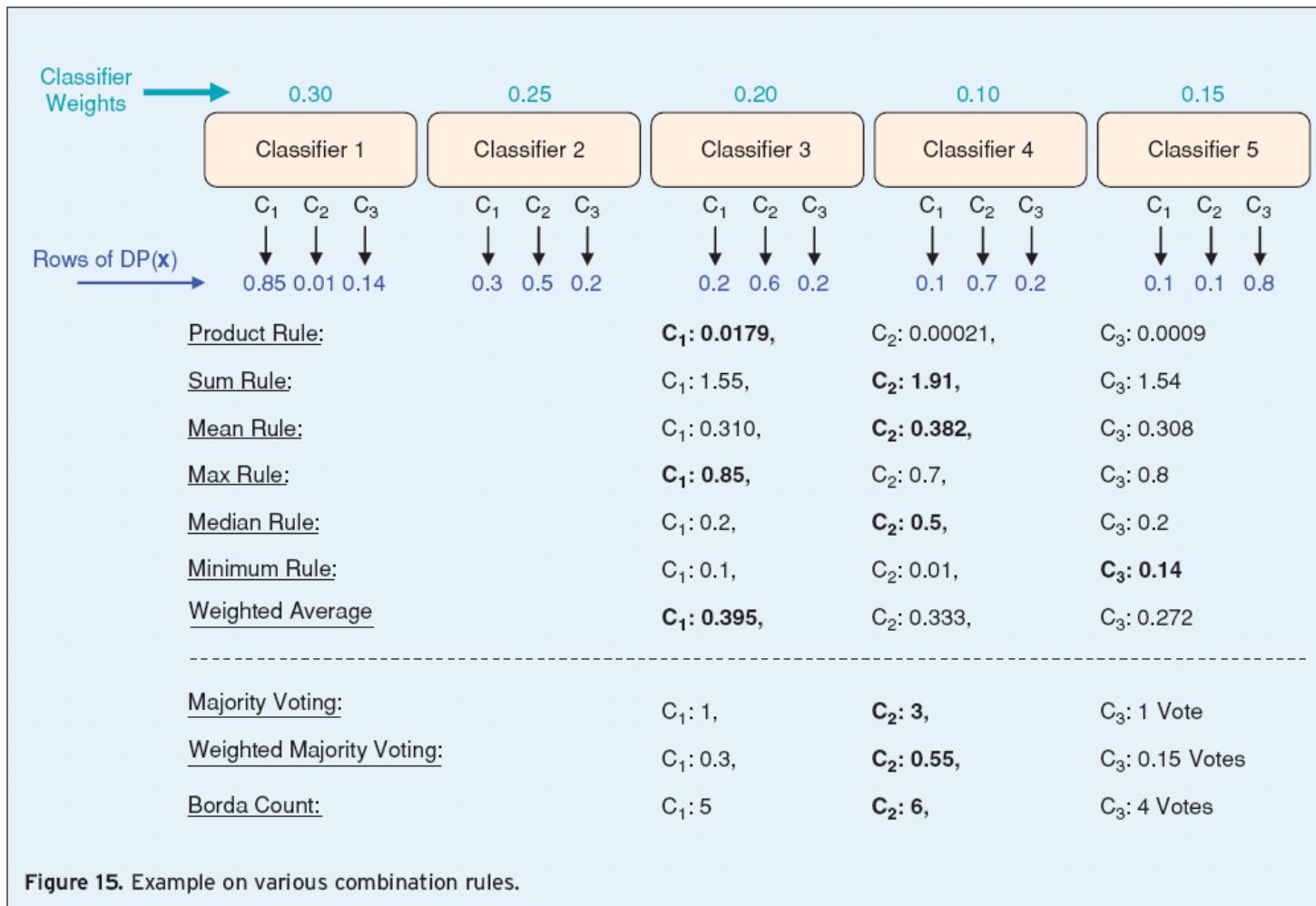
# 배깅: Bagging

- Bagging: Example



# 배깅: Bagging

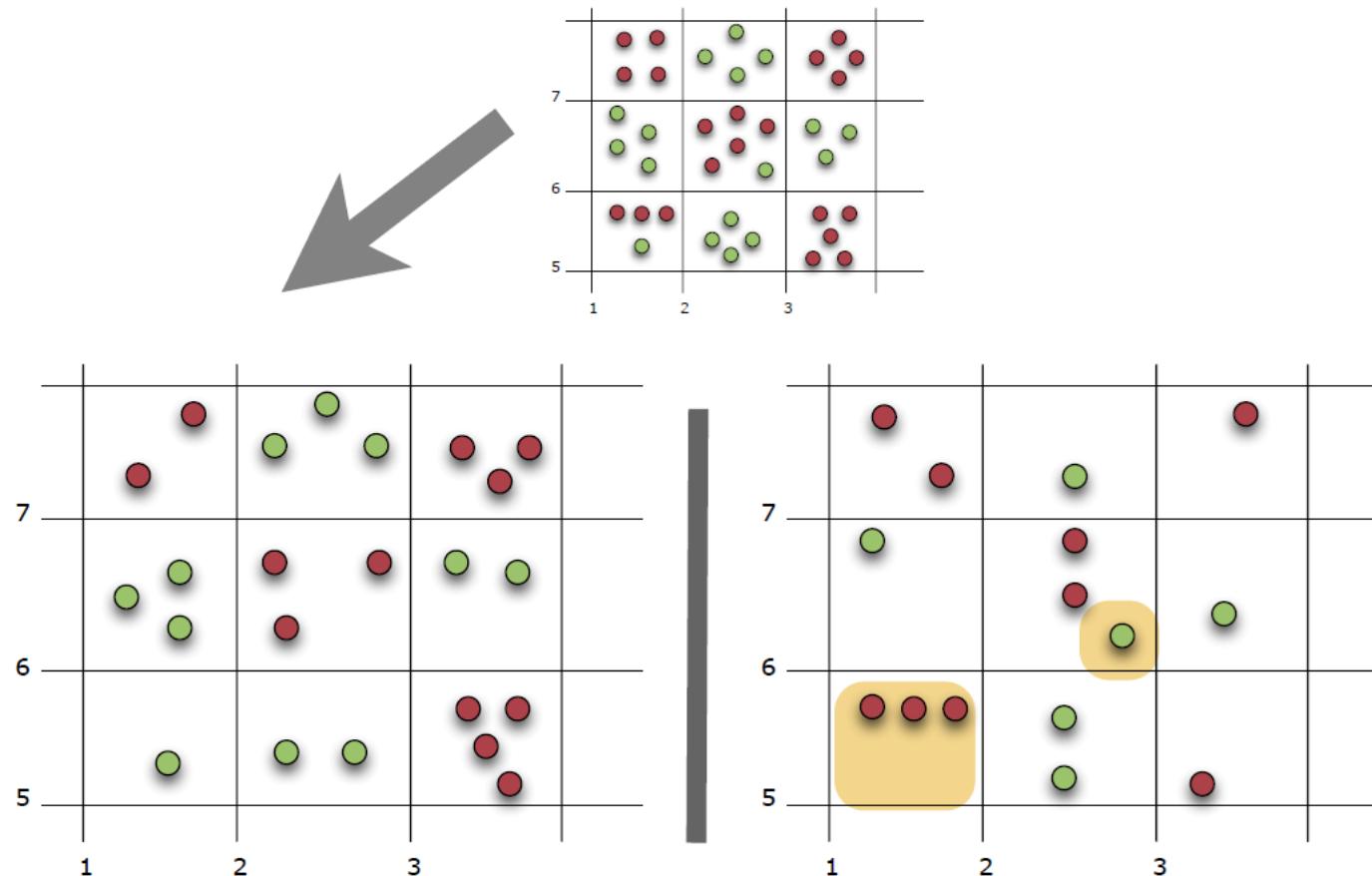
- Bagging: 개별 모델의 결과 취합 방식



# 배깅: Bagging

- Out of bag error (OOB Error)

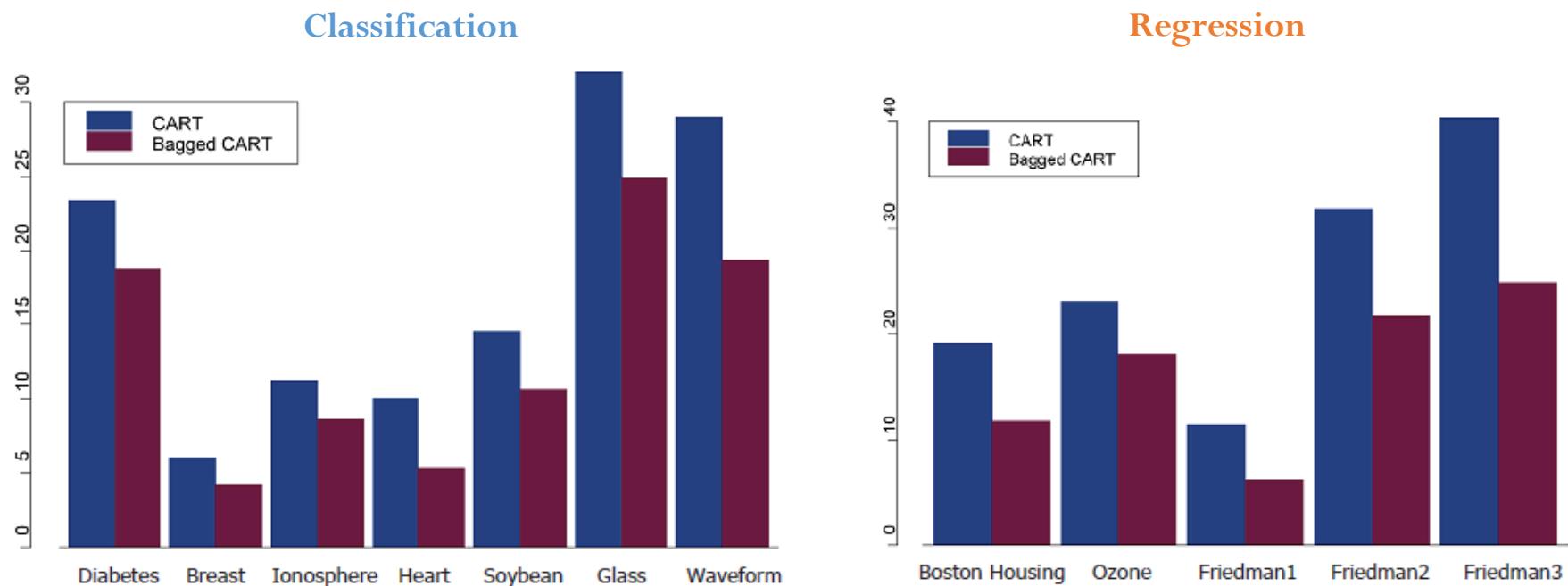
- ✓ 배깅을 사용할 경우, 학습/검증 집합을 사전에 나누지 않고 봇스트랩에 포함되지 않는 데이터들을 검증 집합으로 사용함



# 배깅: Bagging

- 단일 모델과의 비교

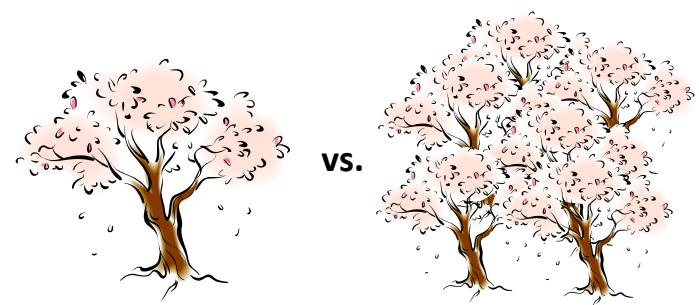
- ✓ 대부분의 경우, 단일 모델에 비해 배깅을 수행하면 모델의 성능이 향상됨을 알 수 있음



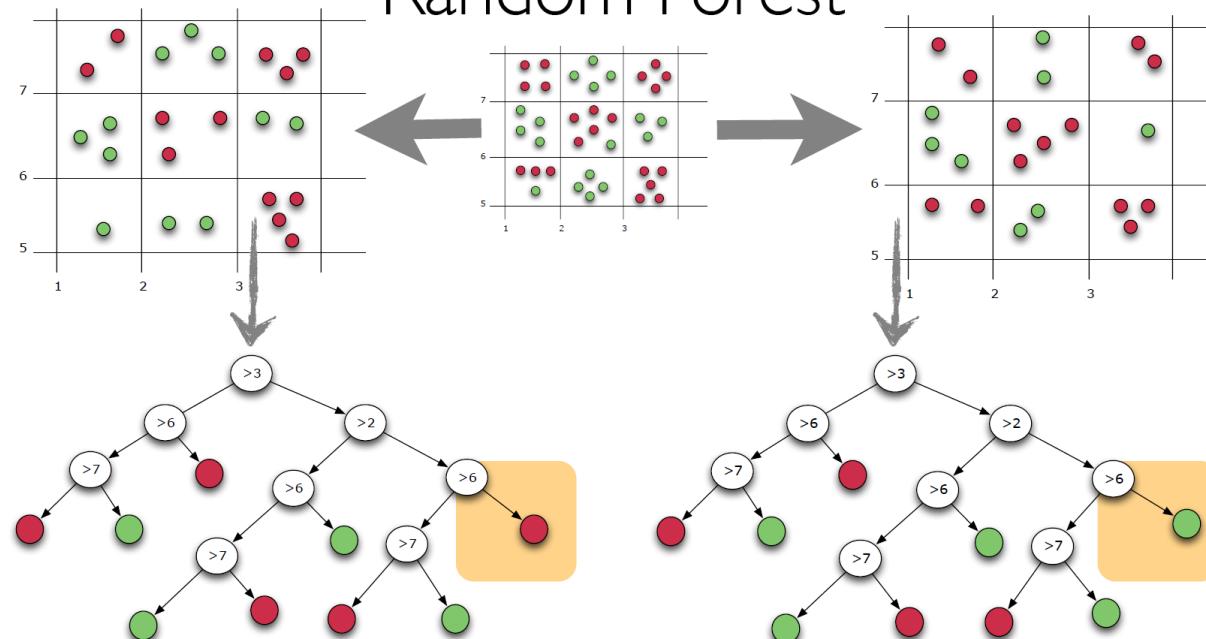
# 랜덤 포레스트: Random Forests

- 랜덤 포레스트

- ✓ 의사결정나무 기반 앙상블의 특수한 형태
  - ✓ 앙상블의 Diversity 확보를 위한 두 가지 장치
    - Bagging
    - Randomly chosen predictor variables



# Random Forest



# 랜덤 포레스트: Random Forests

- Random Forests: Algorithm

1. For  $b = 1$  to  $B$ :
  - (a) Draw a bootstrap sample  $\mathbf{Z}^*$  of size  $N$  from the training data.
  - (b) Grow a random-forest tree  $T_b$  to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size  $n_{min}$  is reached.
    - i. Select  $m$  variables at random from the  $p$  variables.
    - ii. Pick the best variable/split-point among the  $m$ .
    - iii. Split the node into two daughter nodes.
2. Output the ensemble of trees  $\{T_b\}_1^B$ .

To make a prediction at a new point  $x$ :

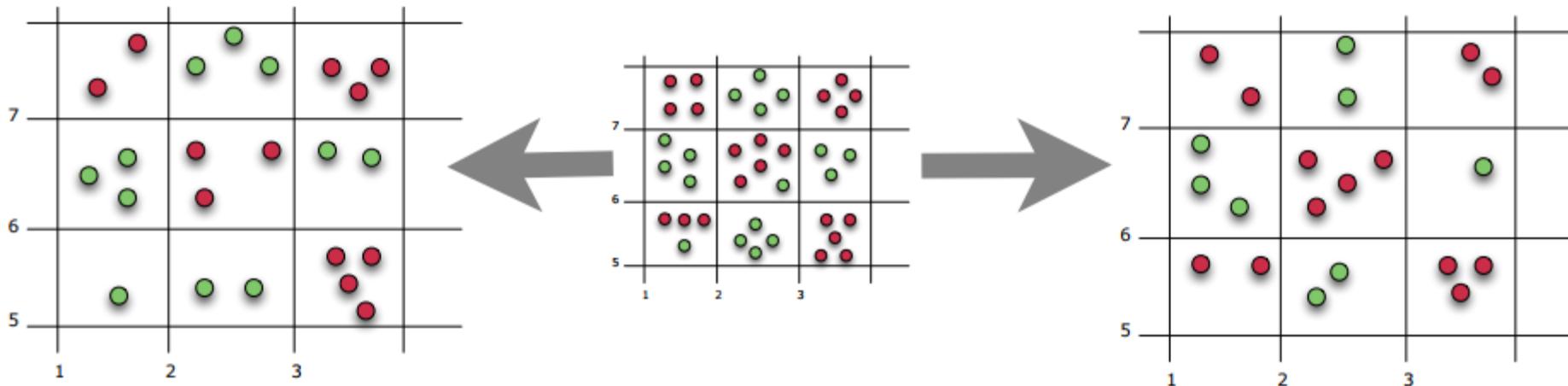
$$\text{Regression: } \hat{f}_{\text{rf}}^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x).$$

*Classification:* Let  $\hat{C}_b(x)$  be the class prediction of the  $b$ th random-forest tree. Then  $\hat{C}_{\text{rf}}^B(x) = \text{majority vote } \{\hat{C}_b(x)\}_1^B$ .

# 랜덤 포레스트: Random Forests

- 다양성 확보 방식 I: Bagging

- ✓ 복원추출 기법으로 원래 학습데이터 개체 수 만큼을 샘플링

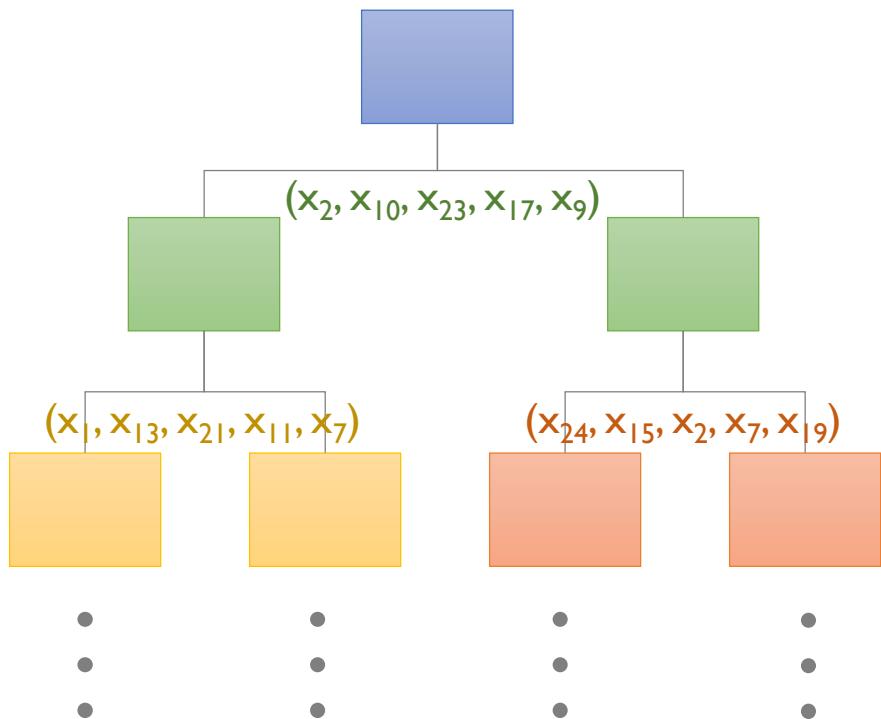
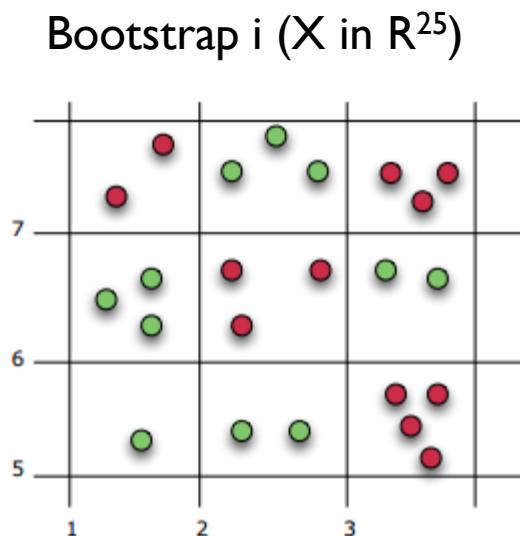


# 랜덤 포레스트: Random Forests

- 다양성 확보 방식 2: 분기 변수의 임의 추출

- ✓ 의사결정나무의 분기점을 탐색할 때, 원래 변수의 수보다 적은 수의 변수를 임의로 선택하여 해당 변수들만을 고려 대상으로 함

- ✓ 예시:



# 랜덤 포레스트: Random Forests

- 일반화 오류

- ✓ 랜덤 포레스트의 개별 트리는 가지치기를 하지 않으므로 과적합의 위험이 있음
- ✓ 양상을 구성모델의 수(population size)가 충분히 클 경우 랜덤 포레스트의 일반화 오류는 다음과 같은 상한을 가짐

$$\text{Generalization Error} \leq \frac{\bar{\rho}(1 - s^2)}{s^2}$$

- $\bar{\rho}$  : 개별 나무들의 예측 결과물 사이의 상관계수 평균
  - $s^2$  : 마진 함수(이범주 분류 문제의 경우 개별 트리의 (정분류율-오분류율)의 평균
- ✓ 개별 나무들이 정확할수록 마진함수가 커지며, 따라서 일반화 오류가 낮아짐
  - ✓ 개별 나무들의 상관관계가 낮을수록 일반화 오류가 낮아짐

# 랜덤 포레스트: Random Forests

- 변수의 중요도

- ✓ 랜덤 포레스트는 다중선형 회귀분석/로지스틱 회귀분석과는 달리 개별 변수가 통계적으로 얼마나 유의한지에 대한 정보를 제공하지 않음
- ✓ 대신 랜덤 포레스트는 다음과 같은 간접적인 방식으로 변수의 중요도를 추정함
  - 1단계: 원래 데이터 집합에 대해서 OOB Error를 구함
  - 2단계: 특정 변수의 값을 임의로 뒤섞은(random permutation) 데이터 집합에 대해서 OOB Error를 구함
  - 3단계: 개별 변수의 중요도는 2단계와 1단계 OOB Error의 평균과 분산을 고려하여 추정

# Random Forests

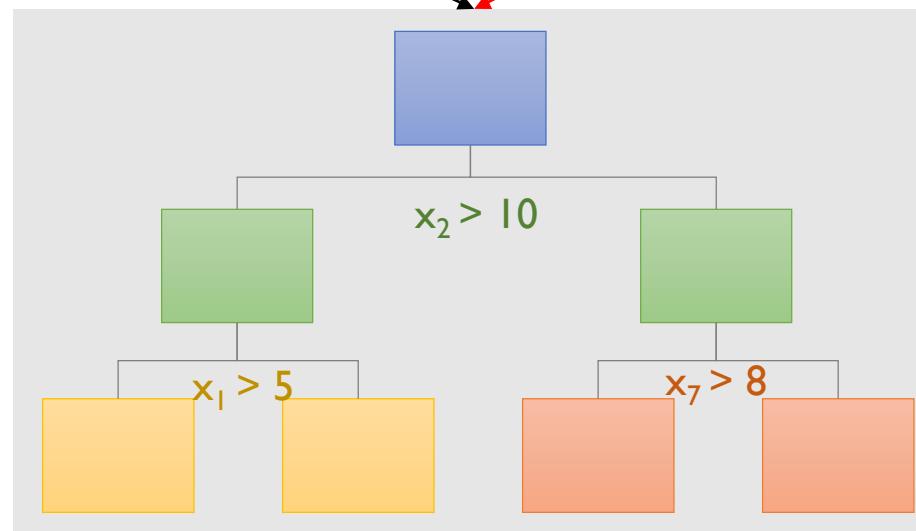
Original OOB Data

ID	X1	...	$X_i$	...	$X_d$	Y
1			0.1			
2			0.5			
3			1.1			
4			1.2			
5			0.4			
6			0.2			
7			0.7			
8			0.8			
9			1.4			
10			1.6			

i번째 변수에 대한 random permutation이 수행된 OOB Data

ID	X1	...	$X_i$	...	$X_d$	Y
1			1.1			
2			0.2			
3			0.1			
4			1.4			
5			1.2			
6			0.5			
7			1.6			
8			0.8			
9			0.7			
10			0.4			

변수 i가 Tree를 split하는데 한번도 사용되지 않았다면



OOB Error of the Original Data  $e_i$



OOB Error of the Permutated Data  $p_i$

# Random Forests

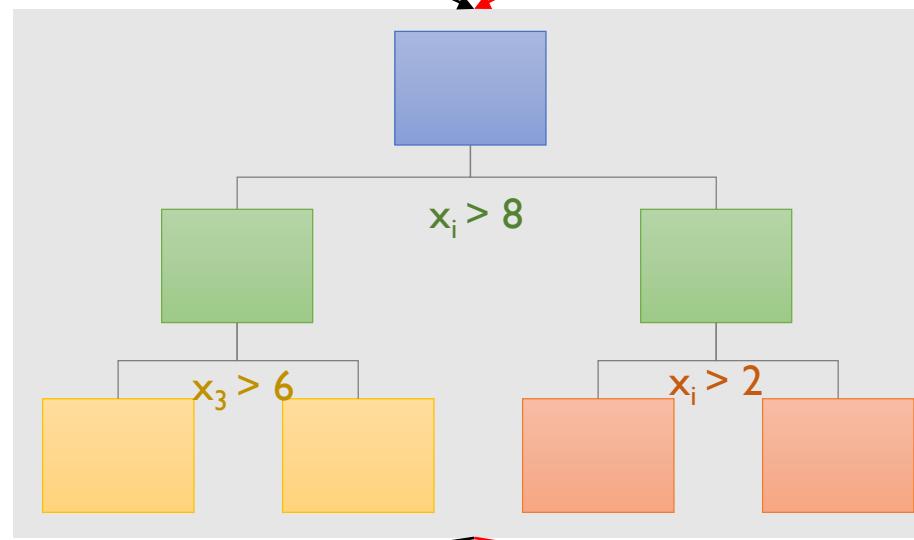
Original OOB Data

ID	X1	...	$X_i$	...	$X_d$	Y
1			0.1			
2			0.5			
3			1.1			
4			1.2			
5			0.4			
6			0.2			
7			0.7			
8			0.8			
9			1.4			
10			1.6			

i번째 변수에 대한 random permutation이 수행된 OOB Data

ID	X1	...	$X_i$	...	$X_d$	Y
1			1.1			
2			0.2			
3			0.1			
4			1.4			
5			1.2			
6			0.5			
7			1.6			
8			0.8			
9			0.7			
10			0.4			

변수  $i$ 가 Tree를 split하는데  
중요하게 사용되었다면



OOB Error of the Original Data  $e_i$

OOB Error of the Permuted Data  $p_i$

# Random Forests

- 변수의 중요도
  - ✓ 랜덤 포레스트에서 변수의 중요도가 높다면
    - 1) Random permutation 전-후의 OOB Error 차이가 크게 나타나야 하며,
    - 2) 그 차이의 편차가 적어야 함
  - m번째 tree에서 변수 i에 대한 Random permutation 전후 OOB error의 차이

$$d_i^m = p_i^m - e_i^m$$

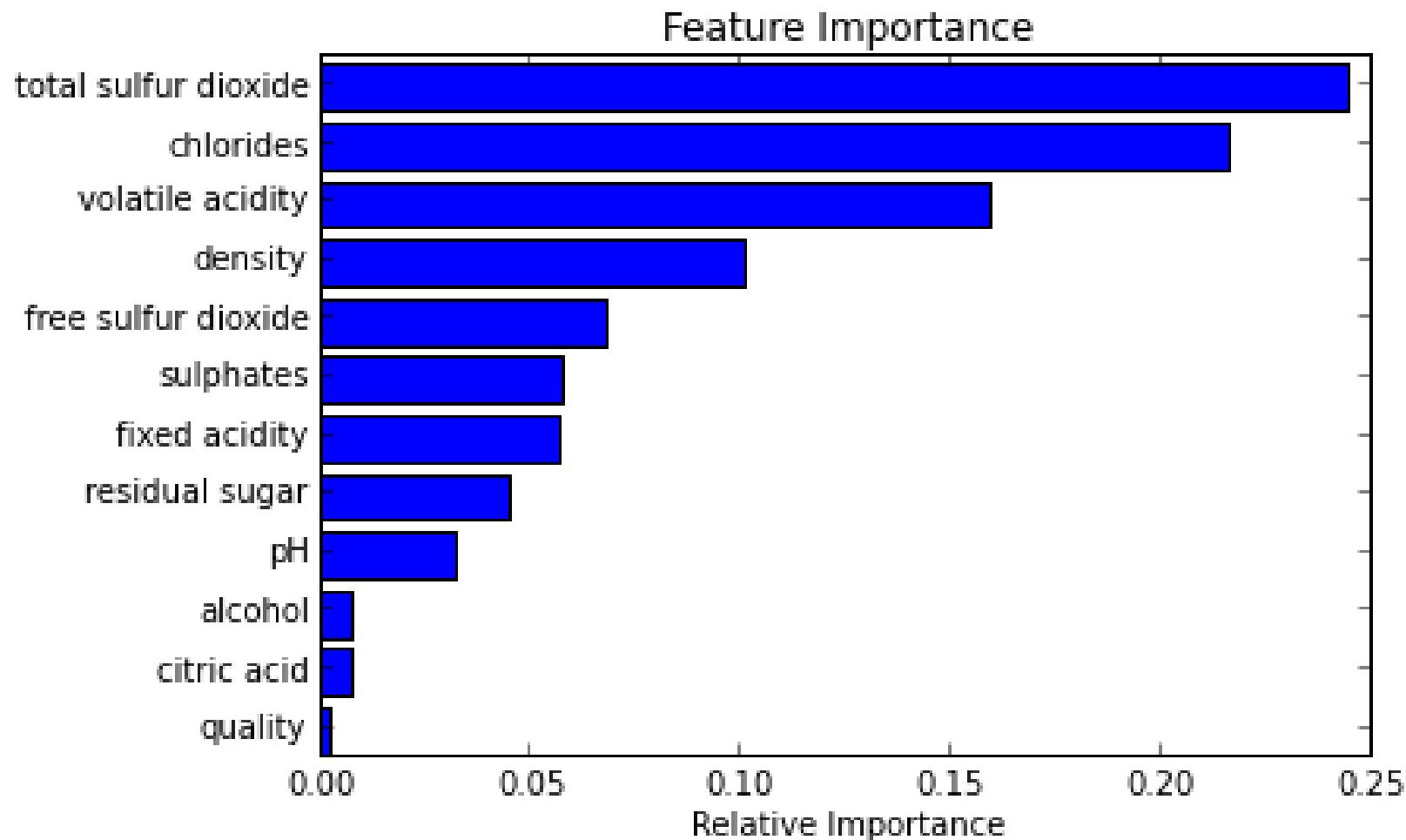
- 전체 Tree들에 대한 OOB error 차이의 평균 및 분산

$$\bar{d}_i = \frac{1}{M} \sum_{m=1}^M d_i^m, \quad s_i^2 = \frac{1}{M-1} \sum_{m=1}^M (d_i^m - \bar{d}_i)^2$$

- i번째 변수의 중요도:  $v_i = \frac{\bar{d}_i}{s_i}$

# 랜덤 포레스트: Random Forests

- 변수 중요도 산출 결과



# AGENDA

- 01 양상블: 배경
- 02 배깅: Bagging & Random Forests
- 03 부스팅: AdaBoost & GBM

# 부스팅:AdaBoost

- AdaBoost: 아이디어

- ✓ Strong model vs. Weak model

- Weak 모델이란 랜덤 모델에 비해 약간의 성능 개선이 있는 모델을 의미하며, 이 모델은 적절한 조치를 취함으로 인해 성능이 우수한 모델로 boosted 될 수 있음

- ✓ 부스팅에서는 현재 모델이 잘 해결하지 못하는 어려운 케이스에 집중

- 1단계: 현재 데이터셋에 대해서 단순한 모델을 이용하여 학습
    - 2단계: 학습 오류가 큰 개체의 선택 확률을 증가시키고 학습 오류가 작은 개체의 선택 확률을 감소시킴
    - 3단계: 앞 단계에서 조정된 확률을 기반으로 다음 단계에서 사용될 학습 데이터셋을 구성
    - 1단계로 되돌아감
    - 최종 결과물은 각 모델의 성능 지표를 가중치로 사용하여 결합

# 부스팅:AdaBoost

- AdaBoost: 알고리즘

---

## Algorithm 2 Adaboost

---

**Input:** Required ensemble size  $T$

**Input:** Training set  $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ , where  $y_i \in \{-1, +1\}$

Define a uniform distribution  $D_1(i)$  over elements of  $S$ .

**for**  $t = 1$  to  $T$  **do**

    Train a model  $h_t$  using distribution  $D_t$ .

    Calculate  $\epsilon_t = P_{D_t}(h_t(x) \neq y)$

    If  $\epsilon_t \geq 0.5$  break

    Set  $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$

    Update  $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

        where  $Z_t$  is a normalization factor so that  $D_{t+1}$  is a valid distribution.

**end for**

For a new testing point  $(x', y')$ ,

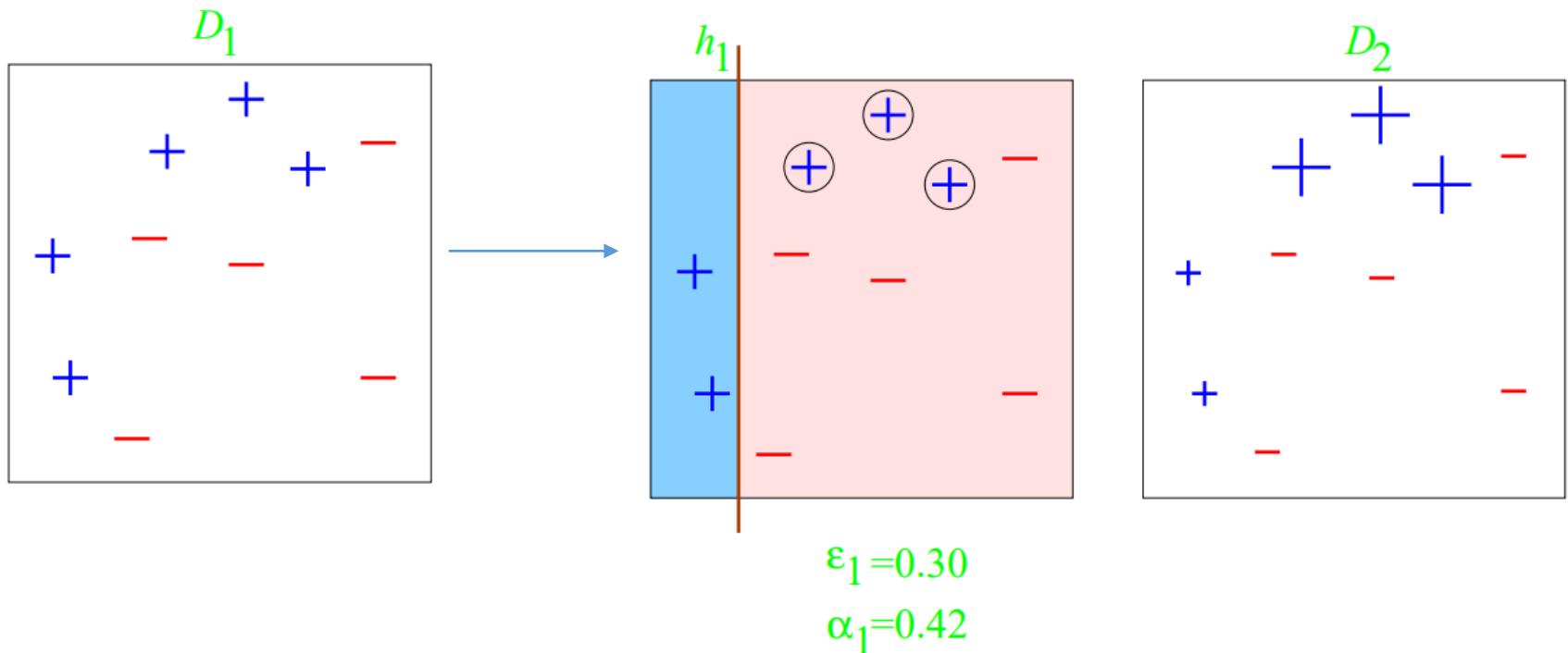
$$H(x') = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x') \right)$$

---

# 부스팅: AdaBoost

- AdaBoost 예시

✓ 1단계



- 10개 중 3개의 오분류:  $\epsilon_i = 0.30$
- 모델의 신뢰도:  $\alpha_i = \frac{1}{2} \log \left( \frac{1 - \epsilon_i}{\epsilon_i} \right) = \frac{1}{2} \log \frac{1 - 0.3}{0.3} = 0.42$

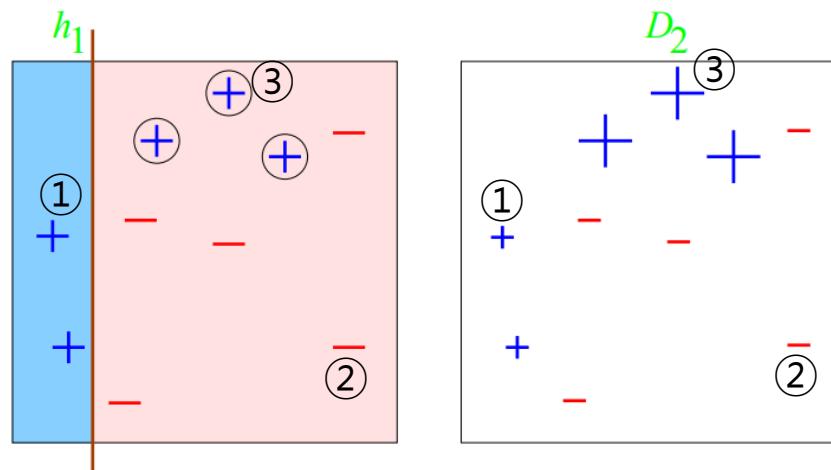
# 부스팅: AdaBoost

- AdaBoost 예시

- 각 개체가 다음 단계의 학습 데이터셋에 선택될 확률

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

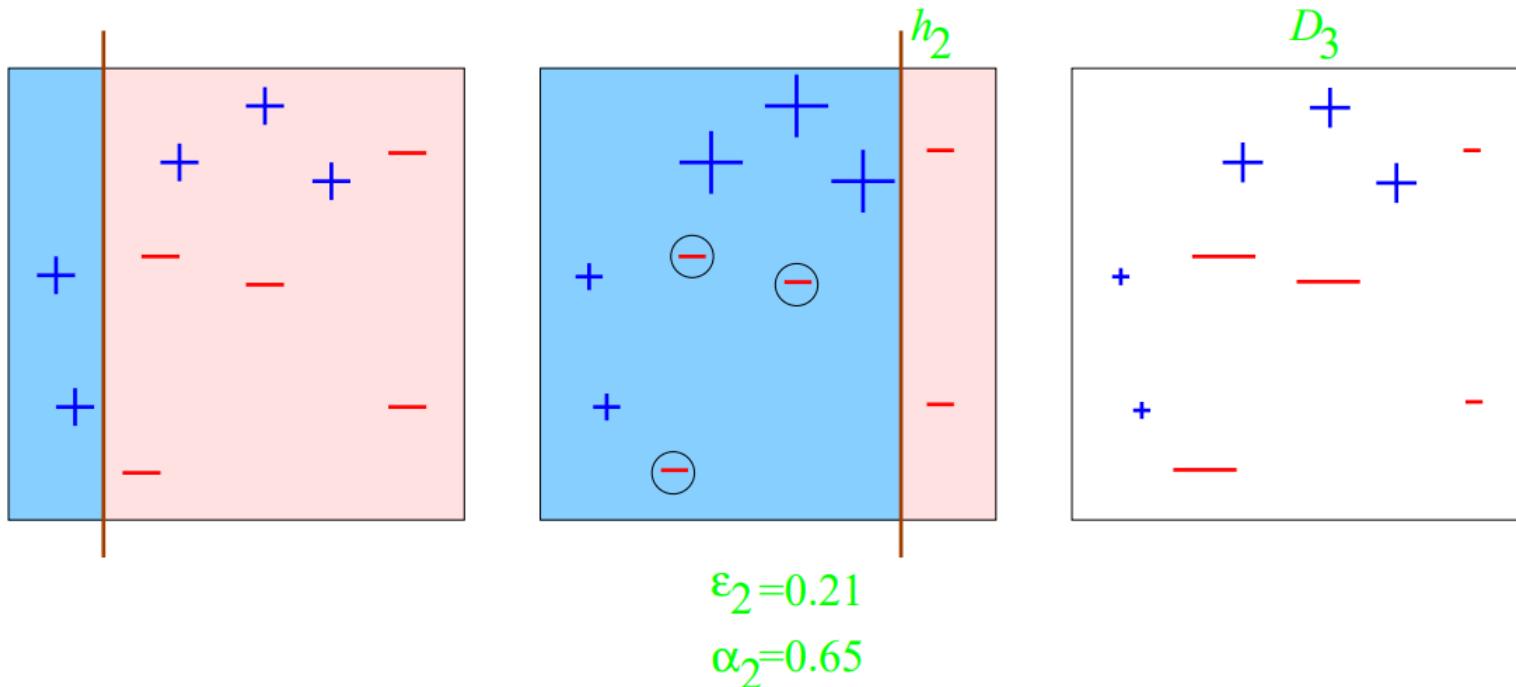
- Case 1:  $y_i = 1, h_t(x_i) = 1 \rightarrow y_i h_t(x_i) = 1 \rightarrow -\alpha_t y_i h_t(x_i) < 0 \rightarrow$  선택확률 감소
- Case 2:  $y_i = -1, h_t(x_i) = -1 \rightarrow y_i h_t(x_i) = 1 \rightarrow -\alpha_t y_i h_t(x_i) < 0 \rightarrow$  선택확률 감소
- Case 3:  $y_i = 1, h_t(x_i) = -1 \rightarrow y_i h_t(x_i) = -1 \rightarrow -\alpha_t y_i h_t(x_i) > 0 \rightarrow$  선택확률 증가
- $\alpha_t$  는 현재 모델의 신뢰도로서 다음단계 선택 확률 증감 폭을 결정



# 부스팅:AdaBoost

- AdaBoost 예시

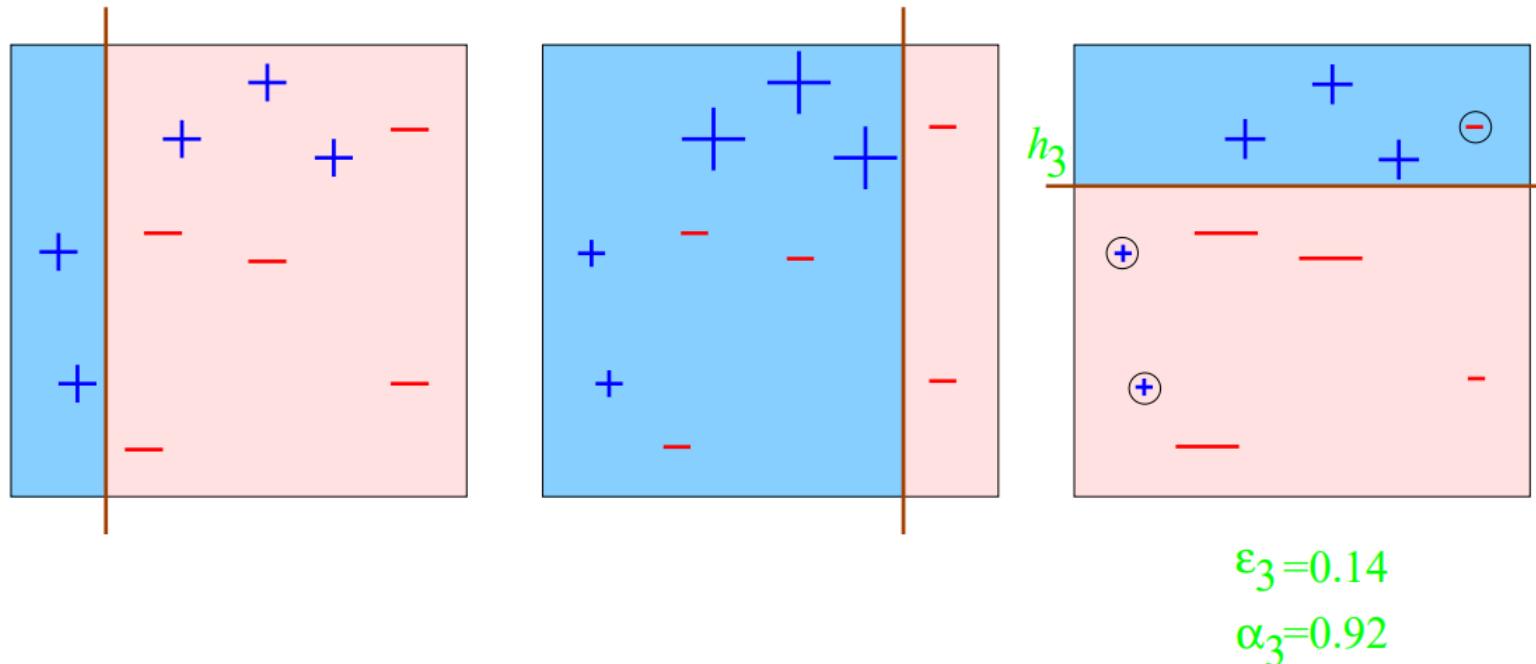
✓ 2단계



# 부스팅:AdaBoost

- AdaBoost 예시

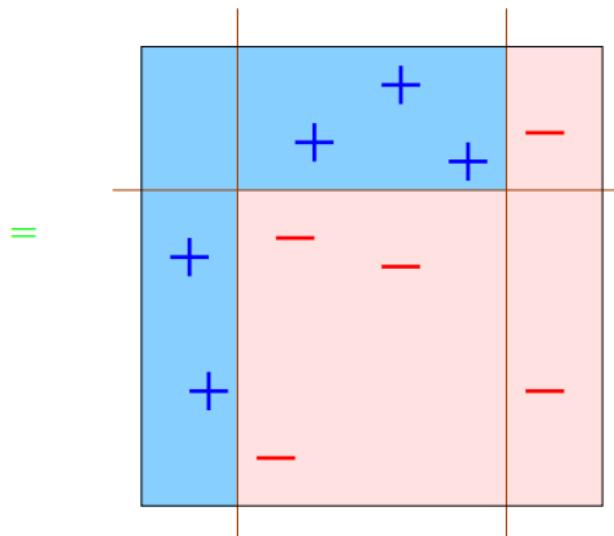
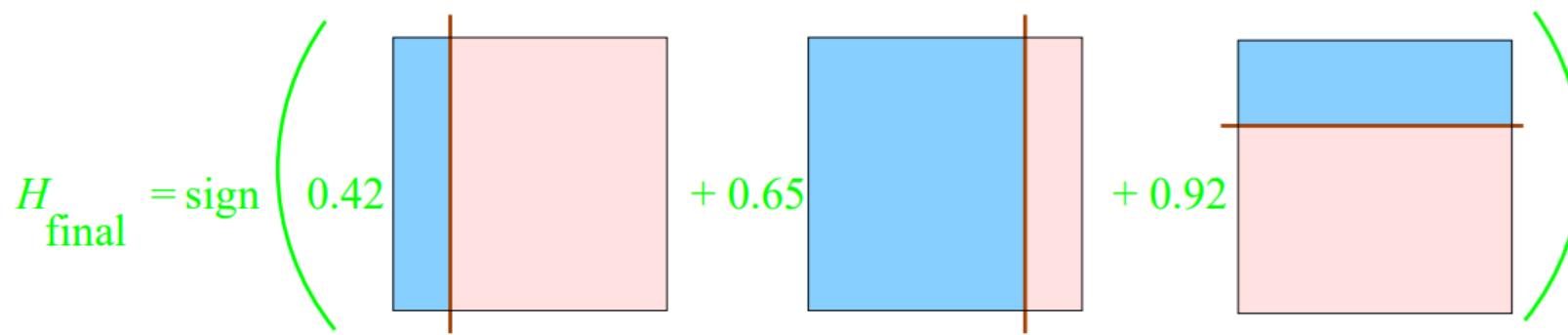
✓ 3단계



# 부스팅:AdaBoost

- AdaBoost 예시

✓ Final Classifier

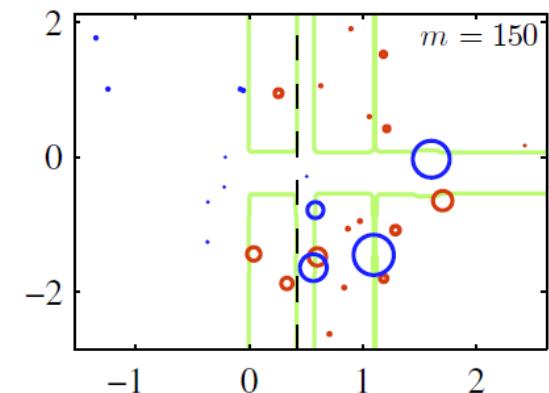
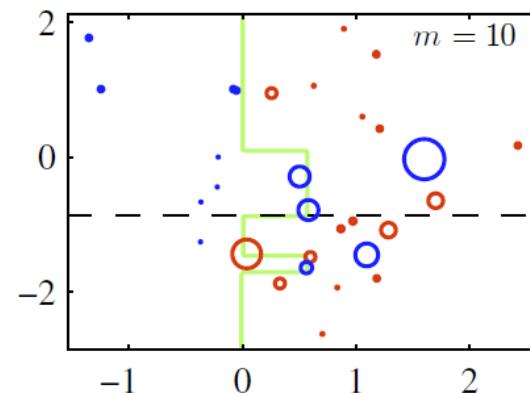
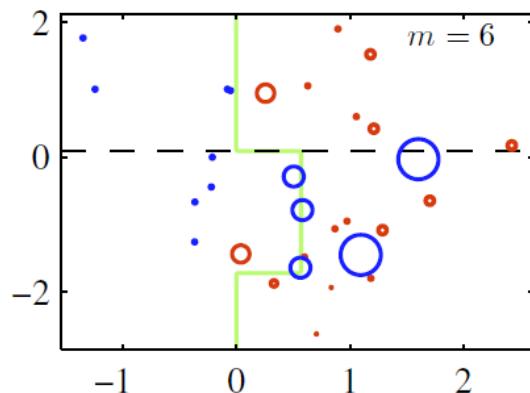
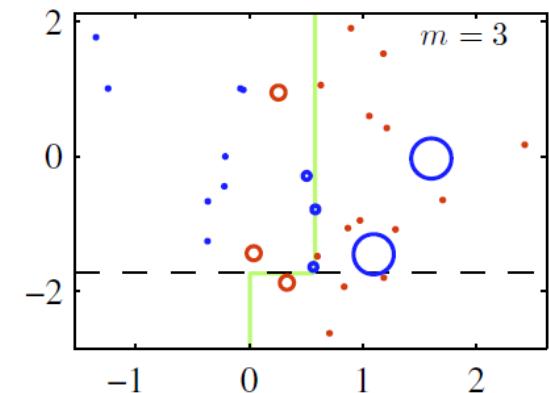
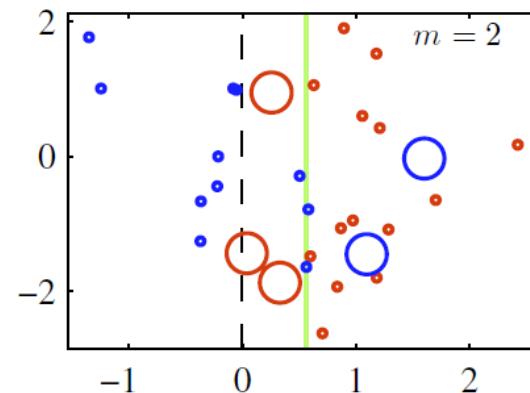
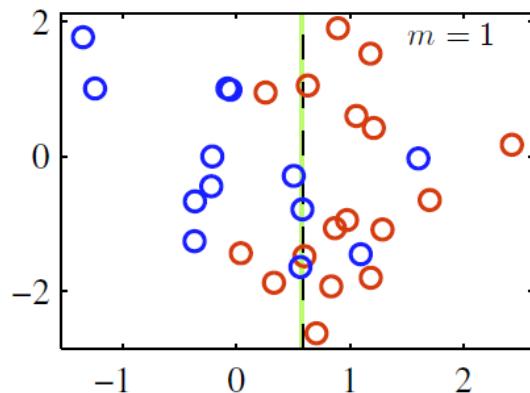


$$H(x') = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x') \right)$$

# 부스팅: AdaBoost

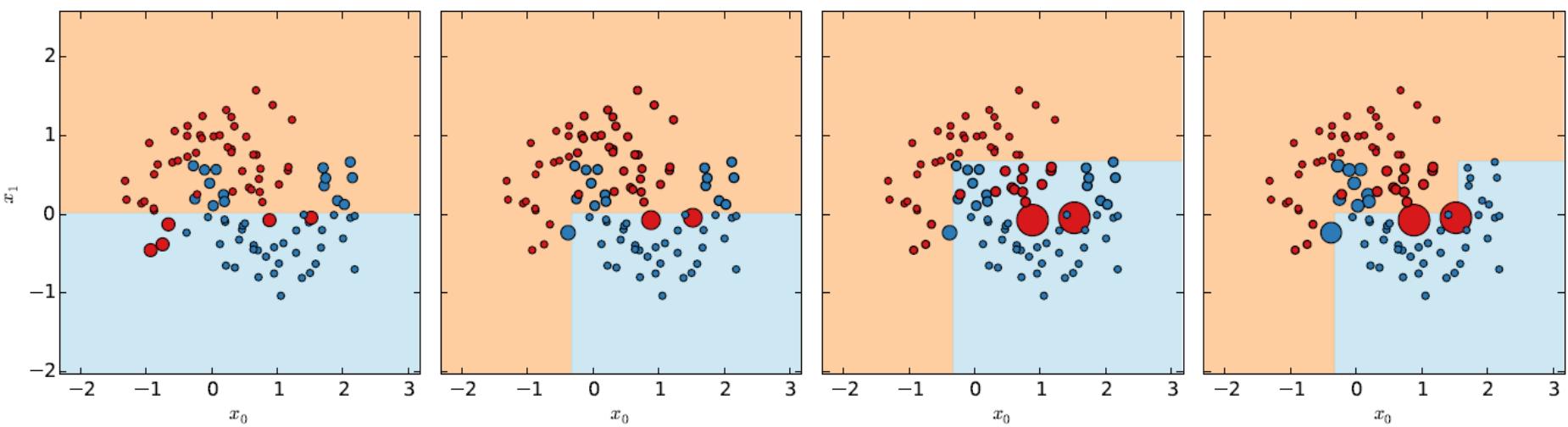
- 부스팅 아이디어: 2범주 분류기, Base learner: Stump Tree

✓ 원의 크기는 해당 개체가 다음 단계의 학습 데이터셋에 선택될 확률을 의미



# 부스팅:AdaBoost

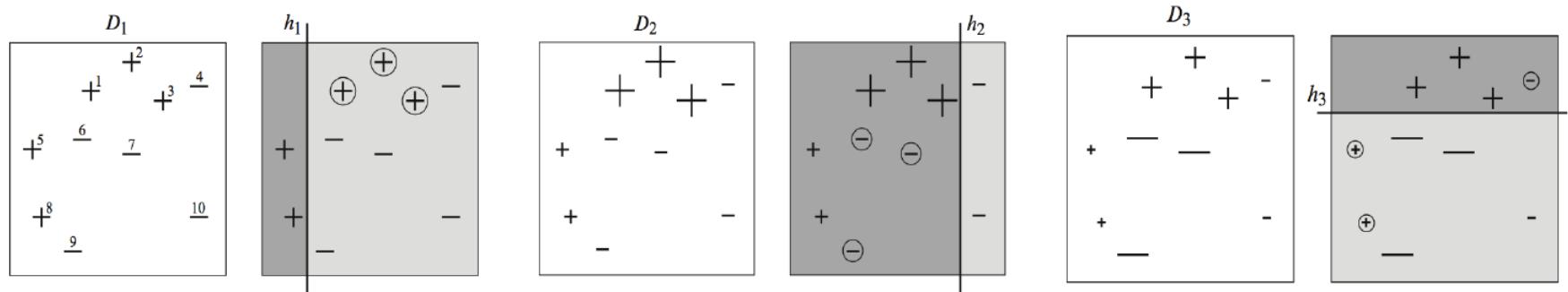
- 부스팅 아이디어: 2범주 분류기, Base learner: Stump Tree
  - ✓ 원의 크기는 해당 개체가 다음 단계의 학습 데이터셋에 선택될 확률을 의미



[https://www.slideshare.net/DataRobot/gradient-boosted-regression-trees-in-scikitlearn?from\\_action=save](https://www.slideshare.net/DataRobot/gradient-boosted-regression-trees-in-scikitlearn?from_action=save)

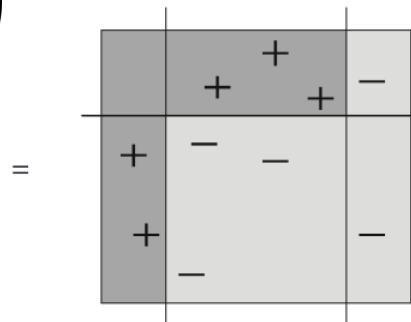
# 부스팅: AdaBoost

- AdaBoost: 알고리즘



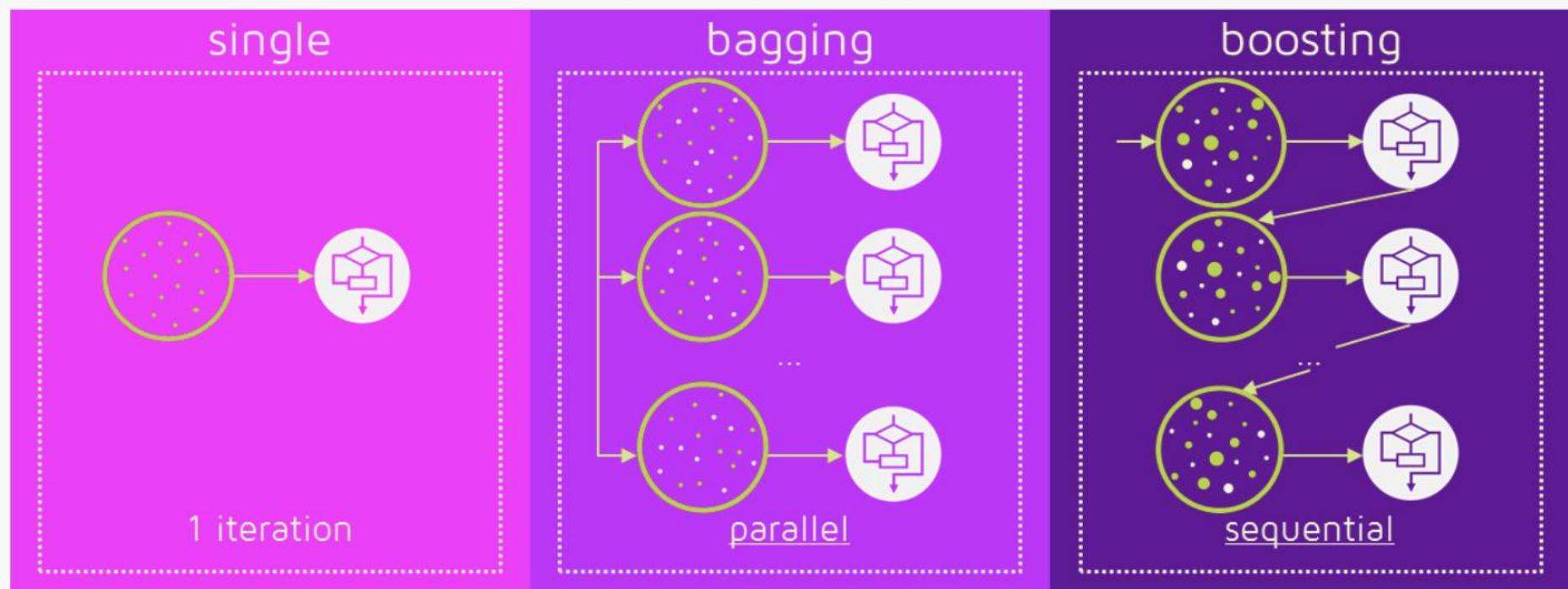
$$H(x') = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x') \right)$$

$$H = \text{sign} \left( 0.42 \begin{array}{|c|c|} \hline \text{+} & \text{-} \\ \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline \text{-} & \text{-} \\ \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline \text{-} & \text{-} \\ \hline \end{array} \right)$$



# 부스팅:AdaBoost

- 단일 모델 vs. 배깅 vs. 부스팅



<https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/>

# 부스팅:AdaBoost

- AdaBoost Example

## AdaBoost in Action

**Kai O. Arras**

Social Robotics Lab, University of Freiburg

Nov 2009  Social Robotics Laboratory

# 부스팅:AdaBoost

- 배깅과 부스팅에서의 개체 선택의 차이

A sample of a single classifier on an imaginary set of data.	
(Original) Training Set	
Training-set-1:	1, 2, 3, 4, 5, 6, 7, 8

A sample of Bagging on the same data.	
(Resampled) Training Set	
Training-set-1:	2, 7, 8, 3, 7, 6, 3, 1
Training-set-2:	7, 8, 5, 6, 4, 2, 7, 1
Training-set-3:	3, 6, 2, 7, 5, 6, 2, 2
Training-set-4:	4, 5, 1, 4, 6, 4, 3, 8

각 학습 집합에서  
개체들이 **무작위로**  
선택됨

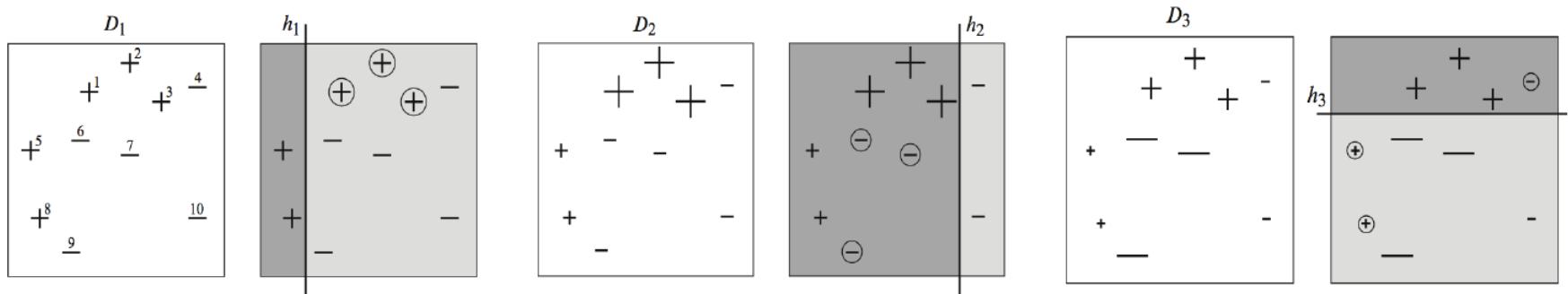
A sample of Boosting on the same data.	
(Resampled) Training Set	
Training-set-1:	2, 7, 8, 3, 7, 6, 3, 1
Training-set-2:	1, 4, 5, 4, 1, 5, 6, 4
Training-set-3:	7, 1, 5, 8, 1, 8, 1, 4
Training-set-4:	1, 1, 6, 1, 1, 3, 1, 5

이전 학습 집합에서 **오류가 큰**  
**개체가 집중적**으로 선택됨

# Gradient Boosting Machine (GBM)

**Gradient Boosting** = **Gradient Descent** + **Boosting**

- Adaboost



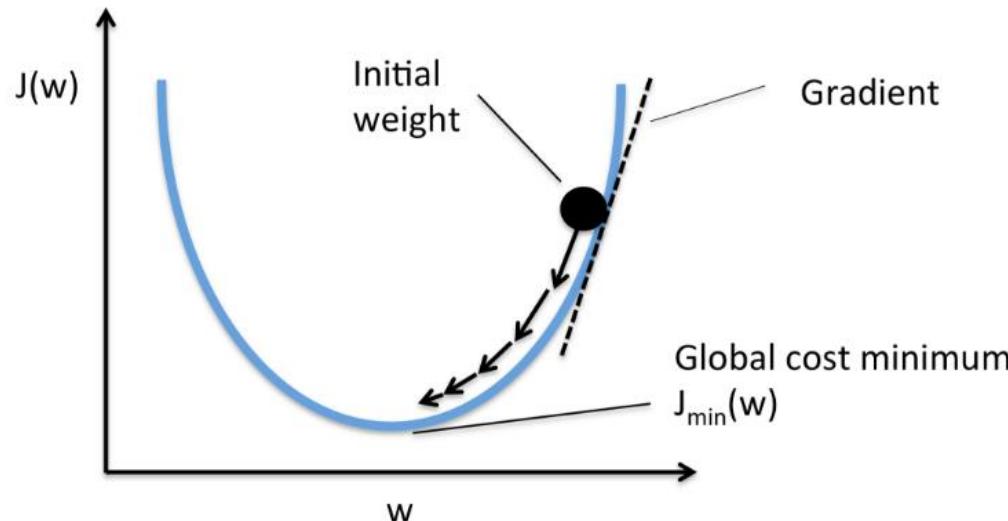
- ✓ 개별 모델을 forward 방식으로 학습
- ✓ 각 단계에서 새로운 Base Learner를 학습하여 이전 단계의 Base Learner의 단점을 보완
- ✓ Adaboost에서는 단점이 각 데이터의 선택 확률에 반영됨

# Gradient Boosting Machine (GBM)

**Gradient Boosting = Gradient Descent + Boosting**

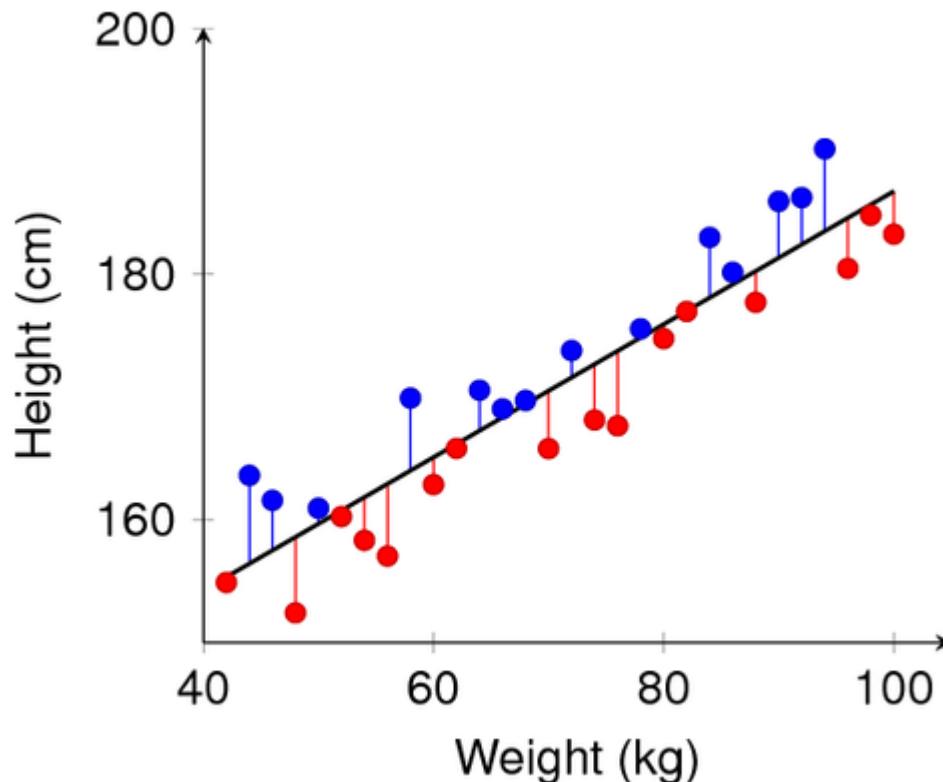
- Gradient Boosting

- ✓ 개별 모델을 forward 방식으로 학습
- ✓ 각 단계에서 새로운 Base Learner를 학습하여 이전 단계의 Base Learner의 단점을 보완
- ✓ Gradient Boosting에서는 단점이 손실 함수(loss function)의 그래디언트(gradient)에 반영됨



# Gradient Boosting Machine (GBM)

- Motivation (for regression problem)
  - ✓ 만일 회귀모형의 잔차를 다음 단계에서 학습하는 모델을 구축할 수 있다면?



# Gradient Boosting Machine (GBM)

- 핵심 아이디어

Original Dataset

$x^1$	$y^1$
$x^2$	$y^2$
$x^3$	$y^3$
$x^4$	$y^4$
$x^5$	$y^5$
$x^6$	$y^6$
$x^7$	$y^7$
$x^8$	$y^8$
$x^9$	$y^9$
$x^{10}$	$y^{10}$

Modified Dataset 1

$x^1$	$y^1 - f_1(x^1)$
$x^2$	$y^2 - f_1(x^2)$
$x^3$	$y^3 - f_1(x^3)$
$x^4$	$y^4 - f_1(x^4)$
$x^5$	$y^5 - f_1(x^5)$
$x^6$	$y^6 - f_1(x^6)$
$x^7$	$y^7 - f_1(x^7)$
$x^8$	$y^8 - f_1(x^8)$
$x^9$	$y^9 - f_1(x^9)$
$x^{10}$	$y^{10} - f_1(x^{10})$

Modified Dataset 2

$x^1$	$y^1 - f_1(x^1) - f_2(x^1)$
$x^2$	$y^2 - f_1(x^2) - f_2(x^2)$
$x^3$	$y^3 - f_1(x^3) - f_2(x^3)$
$x^4$	$y^4 - f_1(x^4) - f_2(x^4)$
$x^5$	$y^5 - f_1(x^5) - f_2(x^5)$
$x^6$	$y^6 - f_1(x^6) - f_2(x^6)$
$x^7$	$y^7 - f_1(x^7) - f_2(x^7)$
$x^8$	$y^8 - f_1(x^8) - f_2(x^8)$
$x^9$	$y^9 - f_1(x^9) - f_2(x^9)$
$x^{10}$	$y^{10} - f_1(x^{10}) - f_2(x^{10})$



$$y = f_1(\mathbf{x})$$

$$y - f_1(\mathbf{x}) = f_2(\mathbf{x})$$

$$y - f_1(\mathbf{x}) - f_2(\mathbf{x}) = f_3(\mathbf{x})$$

# Gradient Boosting Machine (GBM)

- 이 아이디어가 Gradient와 어떻게 연결되는가?

- ✓ 회귀모형의 대표적인 손실 함수 (Squared loss)

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

- ✓ Squared loss의 Gradient

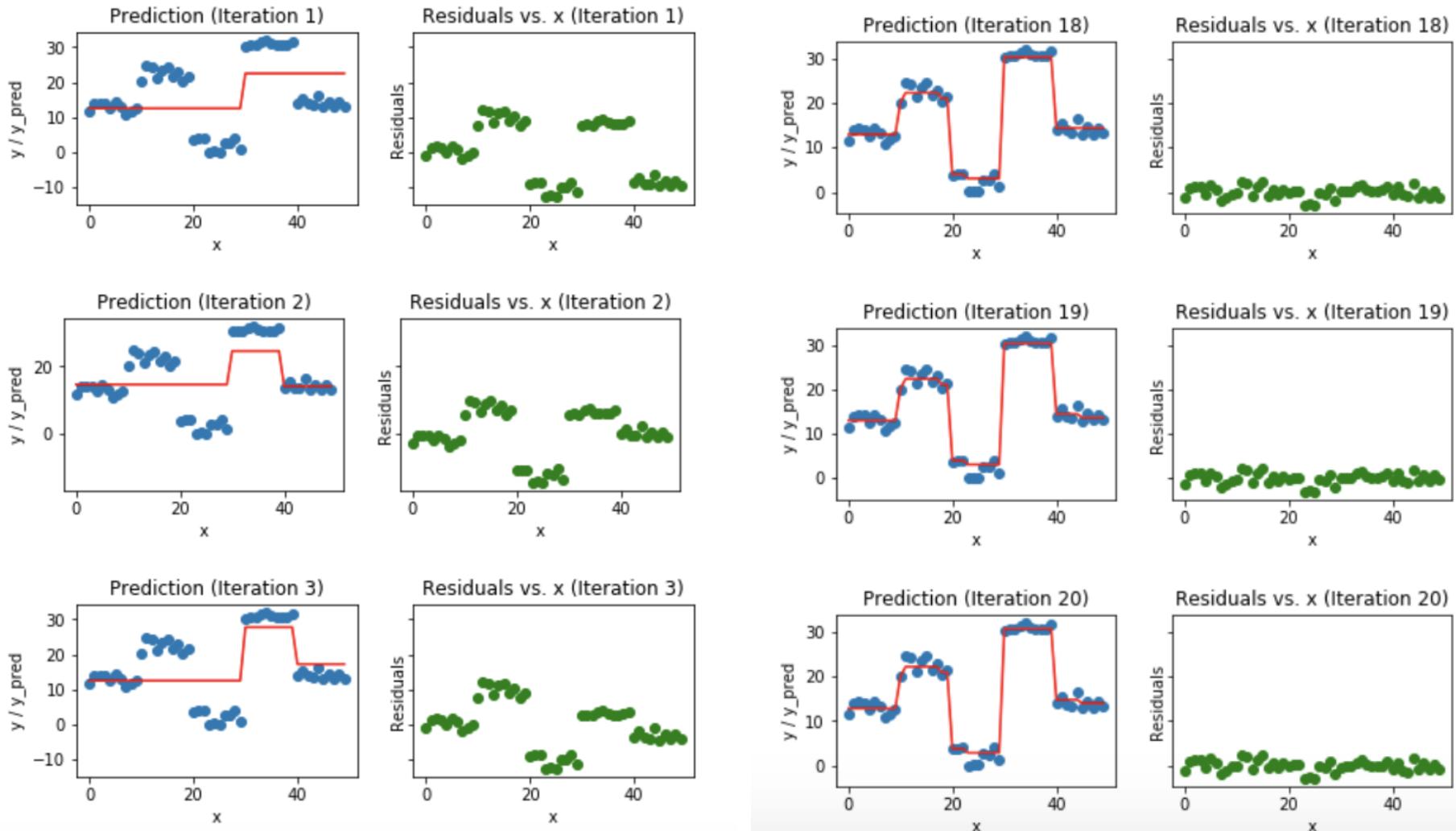
$$\frac{\partial L}{\partial f(\mathbf{x}_i)} = f(\mathbf{x}_i) - y_i$$

- ✓ 회귀모형의 잔차는 Squared loss function의 negative gradient임

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

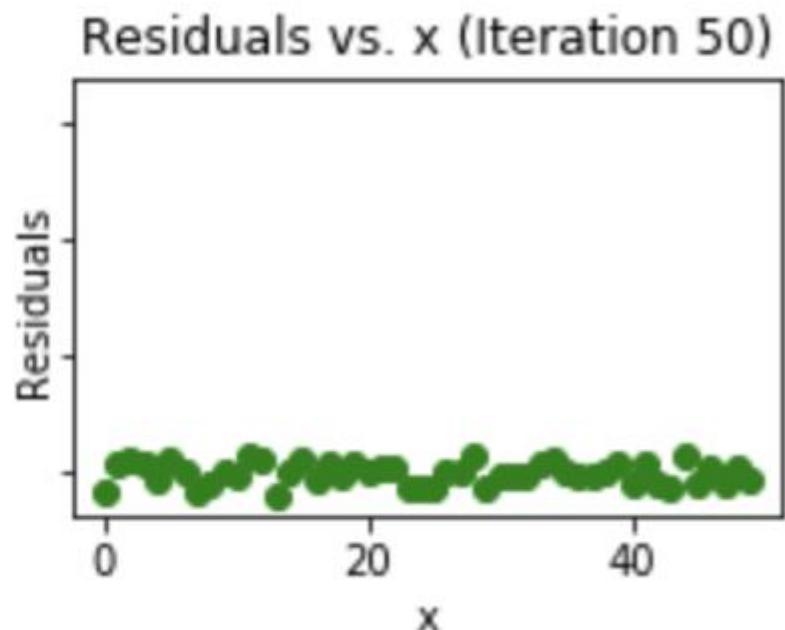
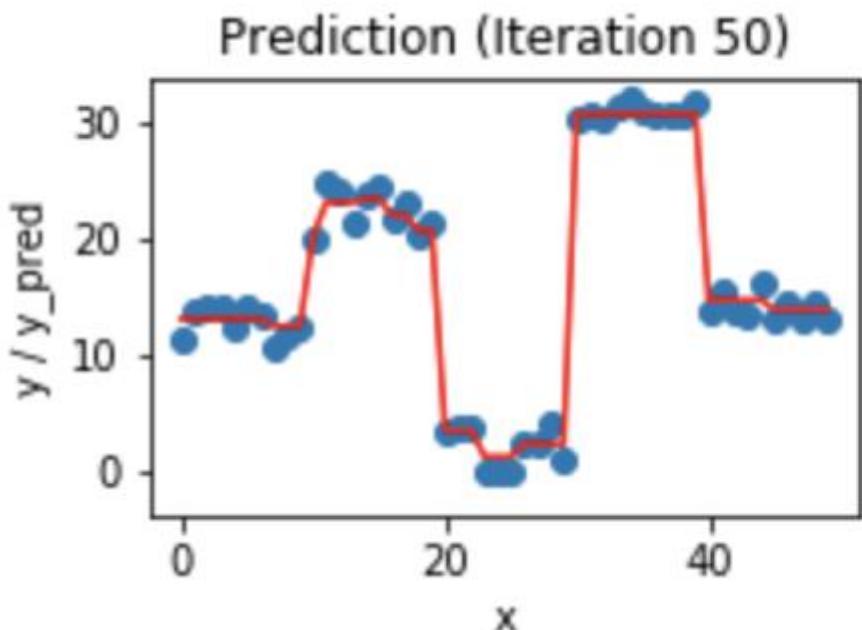
# Gradient Boosting Machine (GBM)

- 회귀모형에 대한 GBM 예시



# Gradient Boosting Machine (GBM)

- 회귀모형에 대한 GBM 예시



<https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d>

# 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  $i = 1, \dots, N$  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  $R_{jm}, j = 1, \dots, J_m$ .

- 2.3 For  $j = 1, \dots, J_m$  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)

- 회귀모형에 대한 손실 함수

## Loss Function

## Formula

Squared loss ( $L_2$ )

$$\Psi(y, f)_{L_2} = \frac{1}{2}(y - f)^2$$

Absolute loss ( $L_1$ )

$$\Psi(y, f)_{L_1} = |y - f|$$

Huber loss

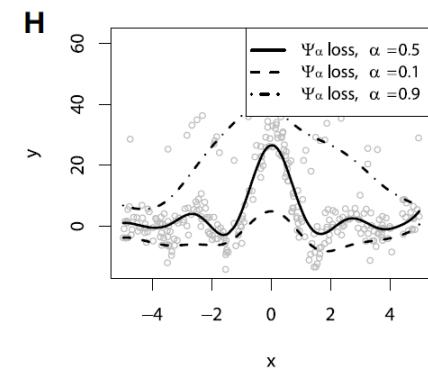
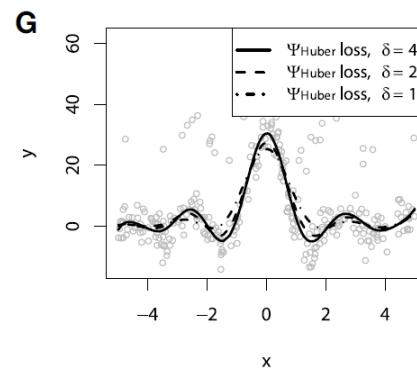
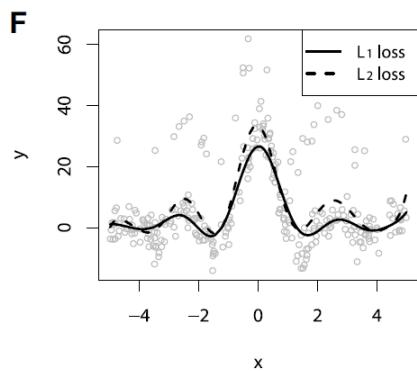
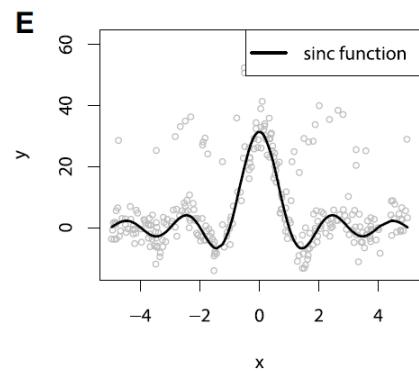
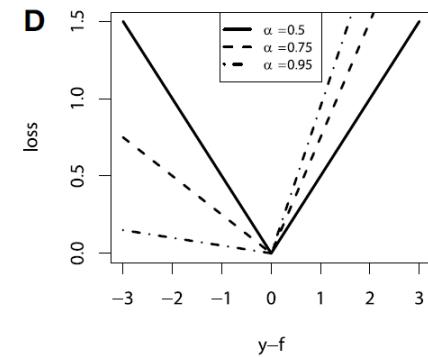
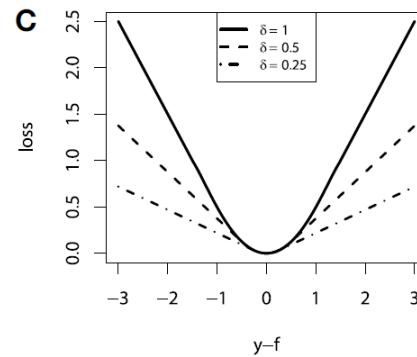
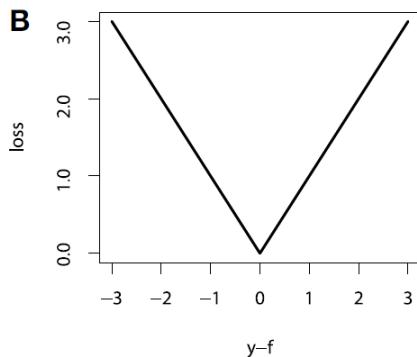
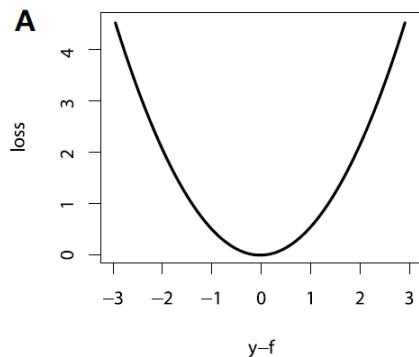
$$\Psi(y, f)_{\text{Huber}, \delta} = \begin{cases} \frac{1}{2}(y - f)^2 & |y - f| \leq \delta \\ \delta(|y - f| - \delta/2) & |y - f| > \delta \end{cases}$$

Quantile loss

$$\Psi(y, f)_\alpha = \begin{cases} (1 - \alpha)|y - f| & y - f \leq 0 \\ \alpha|y - f| & y - f > 0 \end{cases}$$

# Gradient Boosting Machine (GBM)

- 회귀모형에 대한 손실 함수



# Gradient Boosting Machine (GBM)

- 분류 모형에 대한 손실 함수

Loss Function

Bernoulli loss

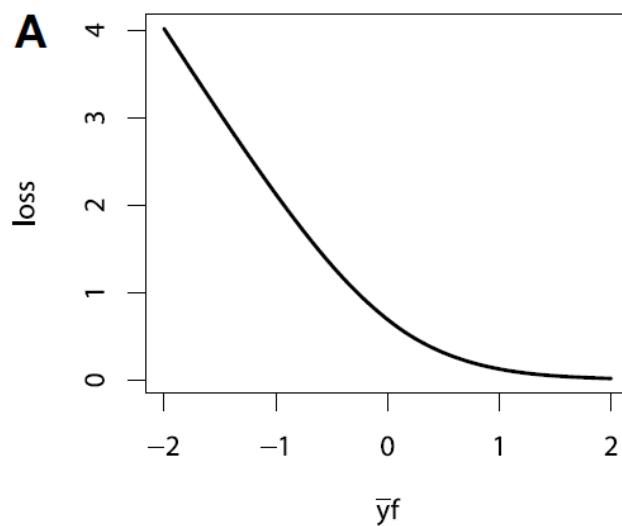
Formula

$$\Psi(y, f)_{\text{Bern}} = \log(1 + \exp(-2\bar{y}f))$$

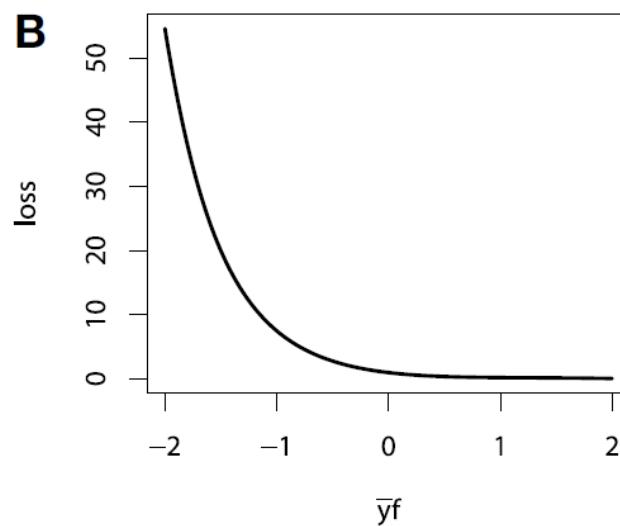
Adaboost loss

$$\Psi(y, f)_{\text{Ada}} = \exp(-\bar{y}f)$$

A

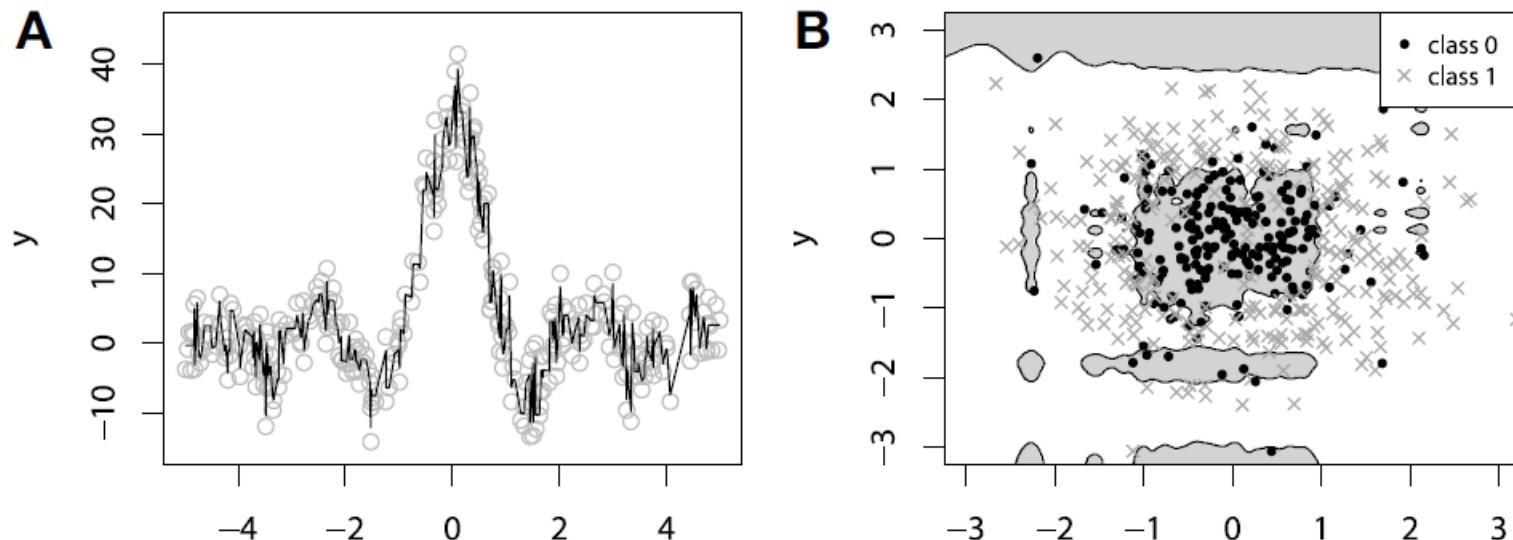


B



# Gradient Boosting Machine (GBM)

- 그래디언트 부스팅의 과적합 방지



- ✓ Subsampling: 학습 데이터의 일부분만 사용 (배깅과는 다르게 비복원 추출 사용)
- ✓ Shrinkage: 다음 모델을 결합할 때 일부러 작은 가중치 사용

$$\hat{f}_t \leftarrow \hat{f}_{t-1} + \lambda p_t h(x, \theta_t)$$

# Gradient Boosting Machine (GBM)

- 변수의 중요도

- ✓  $Influence_j(T)$ : importance of the variable  $j$  in a single tree  $T$ .
- ✓ Assume that there are  $L$  terminal nodes  $\rightarrow L - 1$  splits.

$$Influence_j(T) = \sum_{i=1}^{L-1} (IG_i \times \mathbf{1}(S_i = j))$$

- ✓ Variable importance of Gradient boosting

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



ANY  
questions?