Ensemble Learning

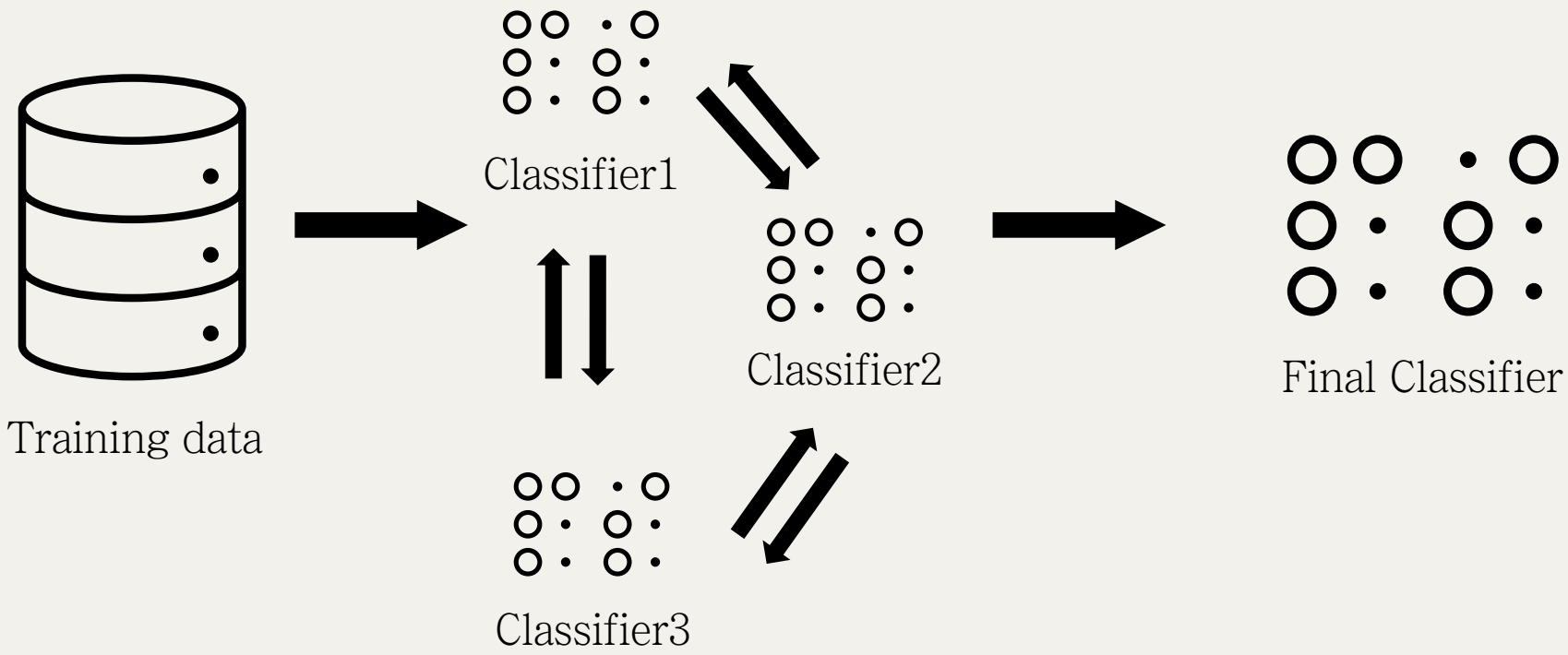
Introduction ~ Boosting

구경민

INDEX

Introduction to Ensemble Learning

Ensemble Learning

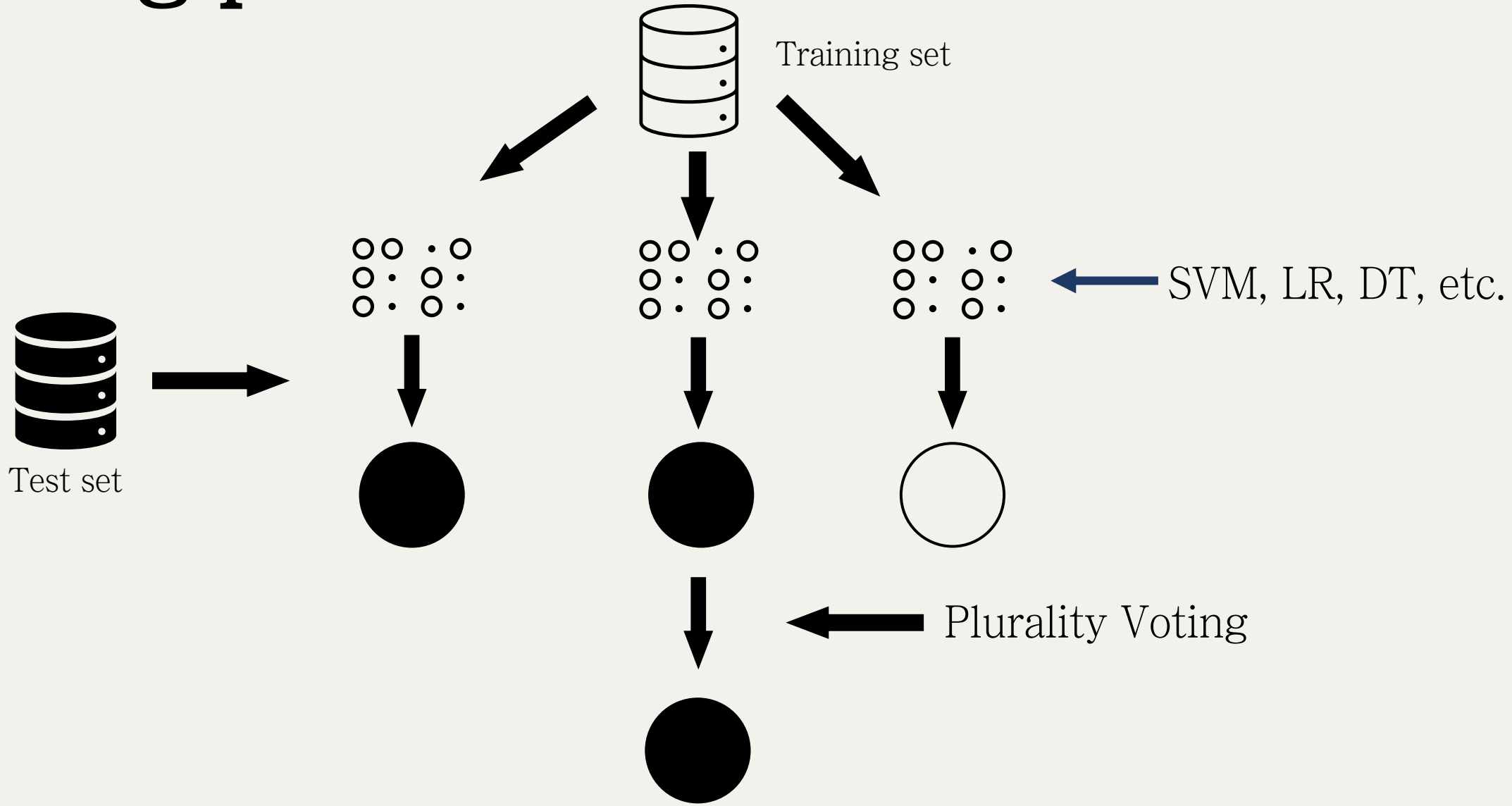


Classifier: 학습 과정에서 만든 개별 분류 모형

-> 여러 분류기를 결합하여 개별 분류기보다 성능이 좋은 최종 분류기를 만드는 과정

Voting

Voting process



Voting process

훈련셋을 통해 각 classifier가 분류

검증셋을 통해 각 classifier가 분류 예측 결과를 내놓음

각 classifier가 예측한 결과를 다수결 투표를 통해 최종 결과를 선정

Why excellent?

Ex) binary problem

K~B(n,p)

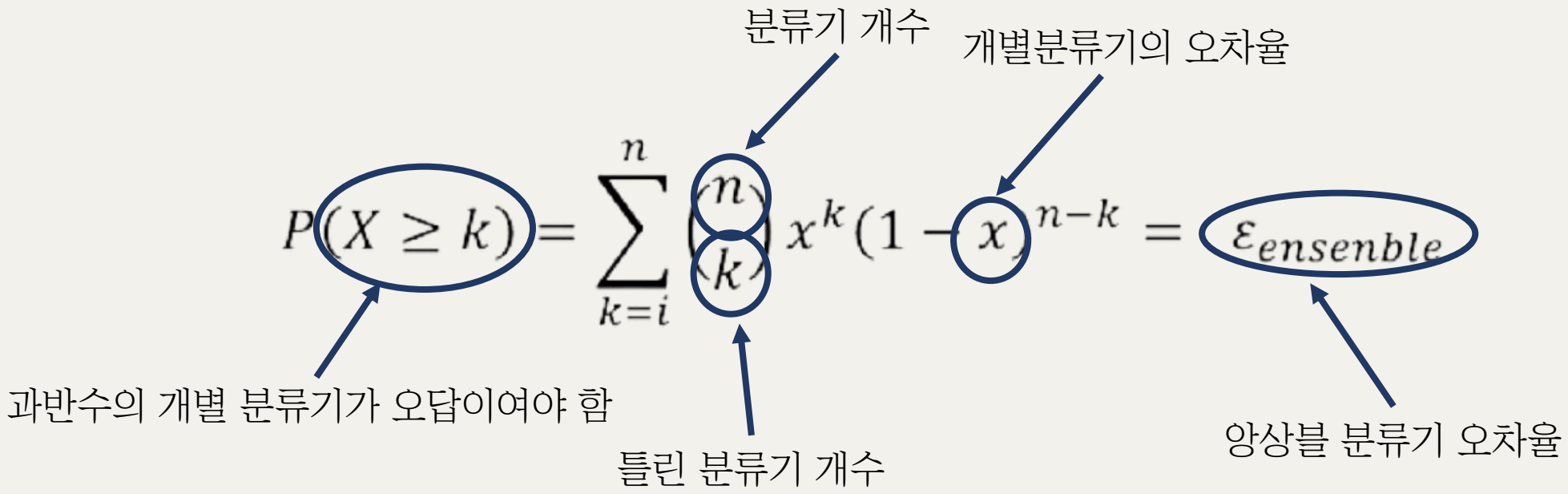
$$\Pr(K = k) = f(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k}$$

(n-k)회의 실패 존재 경우의 수
= 이항계수

성공율이 p인 bernoulli trial이 n번 독립적으로 반복시행 되었을 때,
확률변수 k(성공횟수)의 확률분포를 n과 p를 갖는 분포

Why excellent?

Ex) binary problem



위 식에 값을 대입했을 때 개별 분류기보다 앙상블 분류기의 오차율이 낮다는 것을 알 수 있다

Bagging and Random Forest

Independent Ensemble Method

개별 classifier는 서로 독립적

=

각 classifier는 서로 다른 알고리즘 사용 가능

=

각 classifier는 효과적으로 병렬화 가능

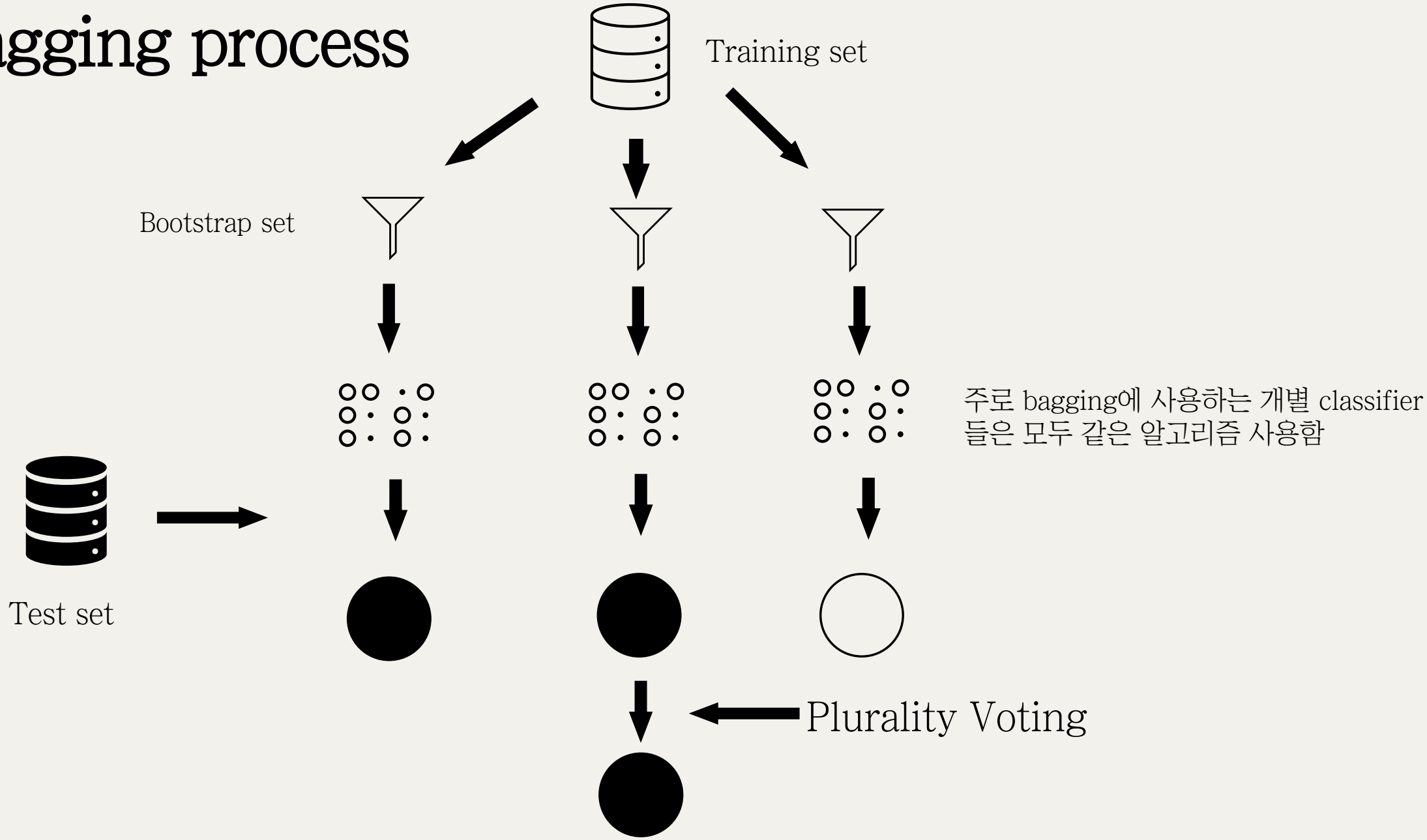
Bagging vs Voting

	Bagging(bootstrap aggregating)	**Voting**
과정	개별 분류기의 분류결과 종합, 최종 분류기 성능 향상	여러 분류기의 결과를 대상으로 최종 label을 정함
최종 결과 산출 방식	다수결 투표	다수결 투표
개별 분류기 다양성	다양x	다양o
데이터 셋 복원추출	가능(bootstrap)	불가능

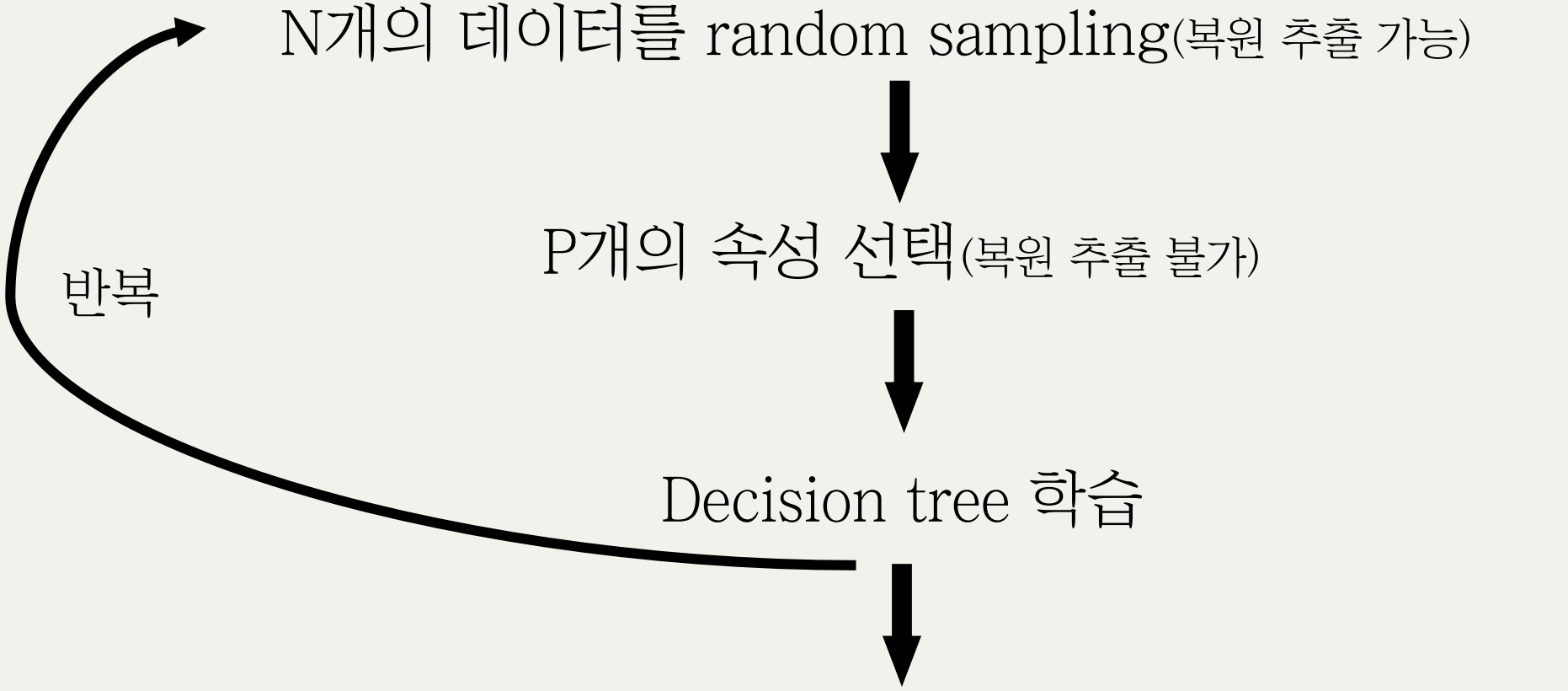
* Bootstrap: 중복을 허용한 랜덤 샘플 방법
Adv. 데이터 양을 임의적으로 늘리고,
데이터 셋의 분포가 고르지 않을 때 고르게 만드는 효과가 있음

Bagging은 Bootstrap을 aggregating하여 학습 데이터가 충분하지 않더라도
충분한 학습효과를 제공하여 높은 편향, 높은 분산으로 인한 underfitting 또는 overfitting 문제를 해결할 수 있음

Bagging process

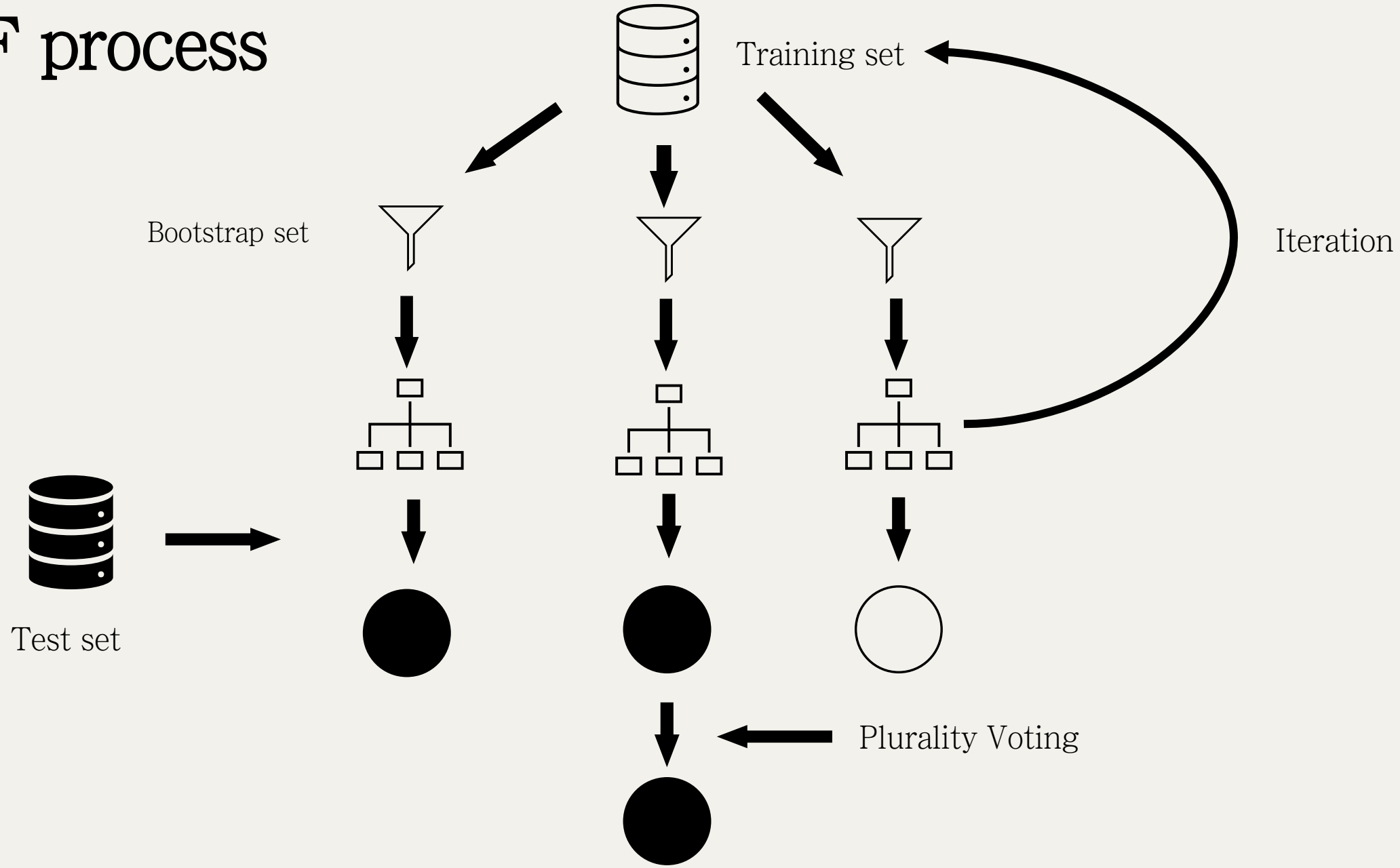


Random Forest process



각 decision tree 별 결과를 다수결로 최종 label 결정

RF process


Training set
Bootstrap set
Iteration
Test set
Plurality Voting

Boosting

Dependent Ensemble Method

개별 classifier는 의존적

=

개별 classifier의 성능에 영향을 받음

Boosting

분류하기 어려운 데이터에 집중한다.

동일한 weighted 할당 → **옳게** 분류된 데이터 weighted 감소

잘못 분류된 데이터 weighted 증가

이전 단계에서 만들어진 classifier는
다음 단계에서 사용할 훈련 데이터의 weighted를 변경하는 데 사용됨

→ 이전 classifier의 성능에 영향을 받음 (의존적 앙상블 방법)

Boosting vs Bagging

	Boosting	Bagging
Classifier 종속 여부	종속적	독립적
데이터 셋 추출	가중치에 비례하여 추출	복원 추출
병렬여부	불가(순차적)	가능

But, Boosting의 classifier는 여러 약한 학습기를 통해 강한 학습기를 만드는 방법으로 낮은 정확도를 보일 수 있다.

AdaBoost

분류하기 어려운 훈련 데이터에 weighted를 더 높임

=

이전에 잘못 분류된 훈련데이터

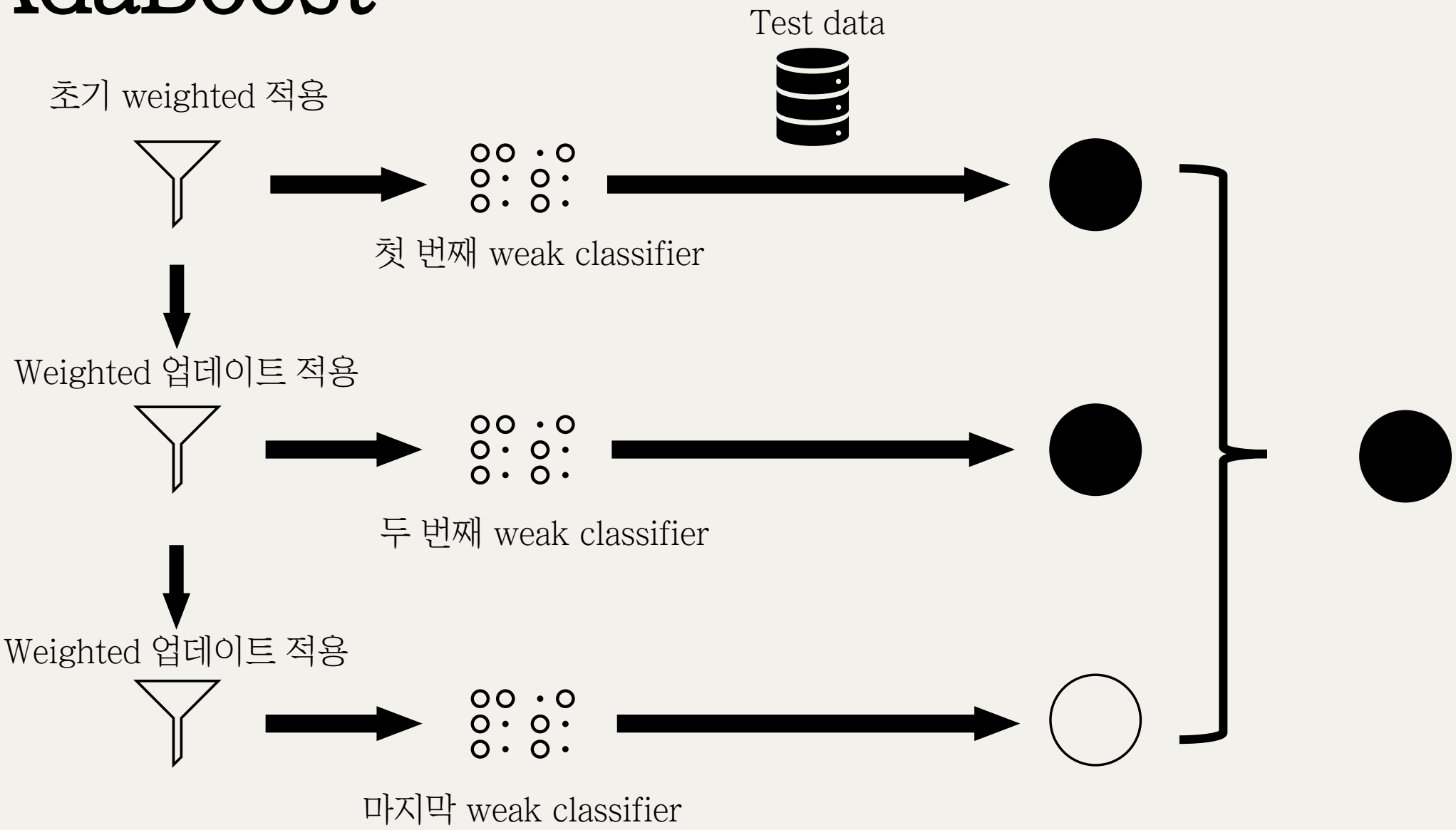
↓

Weighted 증가, error rate 증가

↓

약한 classifier는 이전에 증가한 error rate를 낮추는 방향으로 학습

AdaBoost


Test data
초기 weighted 적용
첫 번째 weak classifier
Weighted 업데이트 적용
두 번째 weak classifier
Weighted 업데이트 적용
마지막 weak classifier

AdaBoost 훈련 데이터의 error rate

Ex) binary problem

초기의 경우 업데이트할 weighted가 없으므로
모든 데이터가 동일한 weighted 적용됨

$$e = \frac{1}{n}\sum_{i=1}^{n} I\left(y_i \neq f_j(x_i)\right)$$

$$\forall I\left(y_i \neq f_j(x_i)\right) = \begin{cases} 0 & y_i \neq f_j(x_i) \\ 1 & \text{O/W} \end{cases}$$

조건 만족 시 1 외 0
-> error rate는 실제 y_i와 weak classifier의
결과가 일치하지 않는 데이터의 개수

$Y \in \{-1,1\}$ = target,

X = feature,

$f_j(X)$ = weak classifier,

F(X) = strong classifier,

n = count of data,

m = count of classifier,

e = error rate

AdaBoost 훈련 데이터의 error rate

Error rate = Expected value of error

$$E\left[I\left(y_i \neq f_j(x_i)\right)\right]$$

Weak classifier의 학습 이후 strong classifier의 최종 예측

$$c(x) = sign\left(\sum_{j=1}^{m} a_j f_j(x)\right)$$

* sign:
이진분류에서
+와 −를 분류하기 위함

데이터 개별 weighted
w_i 를 통해 구할 수 있음

$$a_j = \frac{1}{2}\log\left(\frac{1-e_j}{e_j}\right)$$

-〉 개별 weak classifier의 예측 값에 weighted a_j 를
적용 후 모두 합한 sign(부호)

AdaBoost algorithm

Discrete target

1. 초기 weighted w_i = 1/n
2. J번 째 weak classifier $f_j(x)$이용하여 학습
3. 위에서 사용한 $f_j(x)$의 weighted가 적용된 error rate 구함
4. Weak classifier에 적용될 a_j 구함
5. a_j와 적용중인 w_i 를 이용해 w_i 를 업데이트 ← $w_i <- w_i \exp[-a_j y_i f_j(x_i)]$, i = 1, ⋯, n
6. Weighted normalization ← $w_i <- \frac{w_i}{\sum_{i=1}^{n} w_i}$
7. 2~6 과정을 weak classifier 수만큼 iteration
8. Strong classifier를 이용하여 최종 예측 값 구함

* 투표 시 weak classifier 별로 투표 결과에 weighted 부여

* Weak classifier를 훈련할 때 훈련데이터 전체를 사용함, 훈련 샘플은 iteration 마다 weighted가 다시 부여, 이전 classifier의 실수한 부분을 학습하는 강력한 classifier를 만듦

AdaBoost algorithm

Contineous target

1. 초기 weighted w_i = 1/n
2. w_i 를 훈련데이터에 적용 후 feature 데이터가 주어질 때 class 확률 추정

$$P_j(x) = \hat{P}_w(y = 1|x) \in [0,1]$$

3. 개별 weak classifier를 업데이트

$$f_j(x) = \frac{1}{2}\log\left(\frac{P_j(x)}{1 - P_j(x)}\right) \in R$$

4. 위에서 구한 f와 현재 데이터에 적용 중인 w_i를 이용해 w_i를 업데이트

$$w_i <- \ w_i \ \exp\left[-y_i \, f_j(x_i)\right], \ i = 1, \cdots, n$$

5. Weighted normalization
6. 2~5 단계를 weak classifier 수만큼 iteration
7. Strong classifier이용해 최종 예측 값 구함

$$F(X) = \ sign(\sum_{j=1}^{m} f_j(x))$$

Gradient boosting

비용 함수를 optimization시켜 학습능력 향상

Gradient(기울기)

벡터 미적분학에서 스칼라장의 최대의 증가율을 나타내는 벡터장

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n} \right)$$

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$$

* 스칼라장: 유클리드 공간의 각 점에 스칼라를 대응시킨 것
  벡터장: 유클리드 공간의 각 점에 벡터를 대응시킨 것

Gradient boosting

작을 수록 성능이 좋음

Data (x_n, y_n) Feature x_n -> Predicted value $F(x_n)$ -> Target y_n -> Residual $y_n - F(x_n)$

$$\Longrightarrow F(x_n) + f(x_n) = y_n$$

$$f(x_n) = y_n - F(x_n)$$

F(X)와 y의 차이를 좁힐 수 있는 모형 도입,
y와 F(X)를 이용해 구할 수 있음

f 는 residual $data$ 쌍 $(x_n, y_n - F(x_n))$ 이용해 학습

Gradient boosting

Definition)

Loss func.

$$L\left(y_i, F(x_i)\right) = \frac{1}{2}\left(y_i - F(x_i)\right)^2$$

Obj. to

$$J = \sum_{i=1}^{n} L\left(y_i, F(x_i)\right)$$

Gradient boosting

기존 모형의 기울기

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_{i=1}^{n} L\left(y_i, F(x_i)\right)}{\partial F(x_i)} = L\left(y_i, F(x_i)\right) = F(x_i) - y_i$$

m번째 F(x) 모형

$$F_m(x) = F_{m-1}(x) - \frac{\partial J}{\partial F(x)}$$

⟶ Gradient의 개념과 관련이 있다

Gradient boosting algorithm

$$F_0(x) = \min_{\gamma} \sum_{i=1}^{n} L(y, \gamma)$$

1. Initialization 단계에서 손실함수를 minimize하는 값으로 설정
2. 다음 과정부터 다섯 번째 과정에 해당하는 과정을 M번 반복
3. Target data 값과 m-1번째 모형의 residual을 구한다.

$$e_{m-1} = y - F_{m-1}(x)$$

4. m번째 weak classifier $f_m(x)$를 위에서 구한 m-1번째 모형의 residual e_{m-1}에 fitting
5. m번째 weak classifier에 학습률 η을 적용하여 모형 전체를 업데이트
6. M번 반복한 최종모형 $F_M(x)$ 출력

$$e_{m-1} = y - F_{m-1}(x)$$

References

- 선형대수와 통계학으로 배우는 머신러닝 with 파이썬
- 이항분포 / 위키백과
- 현대통계학 4판
- 1. 앙상블 기법과 배깅, 부스팅, 스태킹 / 데이터 맛집
- gradient / 위키백과

실습은 나중에 따로 만들면 알려드리겠습니다!

Q & A

다 끝나고