

# Lecture 13: Decision Tree and Ensemble Learning

[SCS4049-02] Machine Learning and Data Science

---

Seongsik Park (s.park@dgu.edu)

AI Department, Dongguk University

# Tentative schedule

week	topic	date (수 / 월)
1	Machine Learning Introduction & Basic Mathematics	09.02 / 09.07
2	Python Practice I & Regression	09.09 / 09.14
3	AI Department Seminar I & Clustering I	09.16 / 09.21
4	Clustering II & Classification I	09.23 / 09.28
5	Classification II	(추석) / 10.05
6	Python Practice II & Support Vector Machine I	10.07 / 10.12
7	Support Vector Machine II & Decision Tree and Ensemble Learning	10.14 / 10.19
8	보충수업(Q&A) & <b>Mid-term exam</b>	10.21 / <b>10.26</b>
9	Neural networks	10.28 / 11.02
10	Backpropagation	11.04 / 11.09
11	Convolutional Neural Network	11.11 / 11.16
12	Model Optimization	11.18 / 11.23
13	Recurrent Neural network	11.25 / 11.30
14	Autoencoders	12.02 / 12.07
15	<b>Final exam</b>	(휴강) / <b>12.14</b>

# Decision Tree

---

## Decision trees의 특징

- Feature 공간을 여러 개의 단순한 영역으로 분할/계층화하는 것
- Feature 공간을 분할하는 일련의 규칙들은 tree의 형태로 표현
- 간단하면서 이해/해석이 용이
- Classification 뿐만 아니라 regression에도 사용이 가능
- Numeric feature와 categorical feature 모두 처리 가능
- 스케일링, 중앙화 등 같은 데이터에 대한 pre-processing이 거의 필요 없음
- 탐색적 데이터 분석(exploratory data analysis)에서 feature 선정 등에 유용
- 예측 정확도 측면에서 다른 머신 러닝 알고리즘보다 떨어지는 경우가 많음
- Bagging, boosting, random forests 등을 사용하면 tree의 예측 정확도를 크게 향상시킬 수 있음

# Training and visualizing a decision tree

## Iris dataset

- 머신 러닝의 지도 학습용 데이터셋으로 널리 알려짐
- 3개의 클래스 (iris setosa, iris versicolor, iris virginica) 별로 각각 50개의 사례
- Feature의 수: 4개

## Terminology

- Row: observation (sample, example, instance, record)  
 $\implies (\mathbf{x}^{(i)}, y^{(i)})$
- Column except the last: feature (predictor, attribute, independent variable, input)  $\implies \mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots, x_D^{(i)})$
- Last column: response (target, outcome, label, ground truth)  
 $\implies y^{(i)}$

# Tree terminology

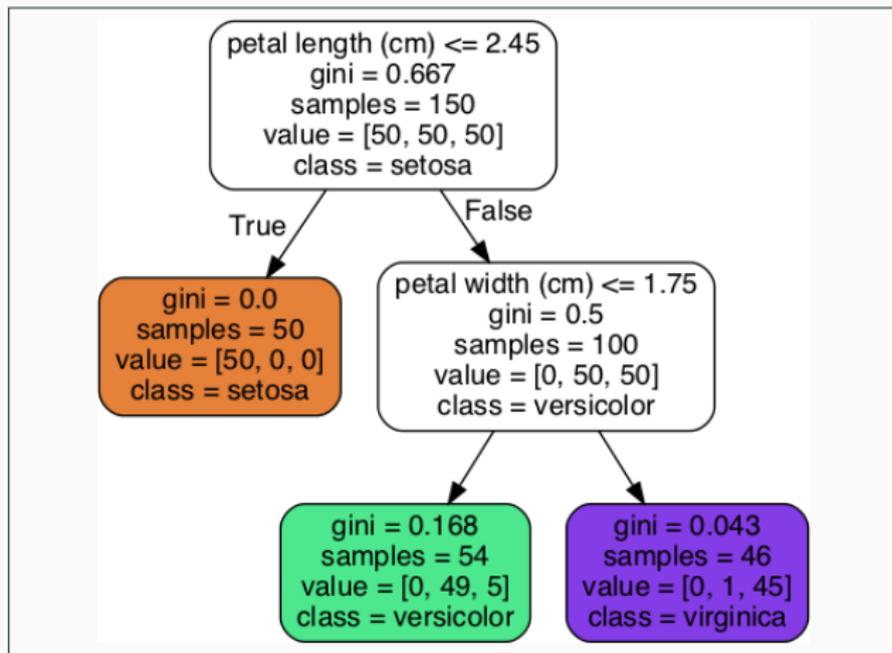
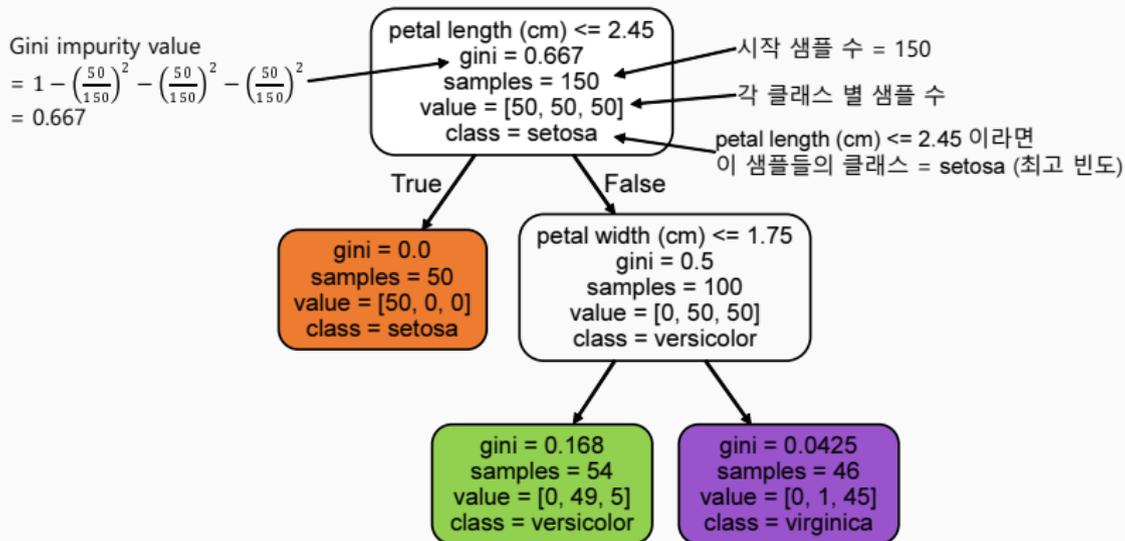


Figure 6-1. Iris Decision Tree

# Prediction flow



# Decision tree with different tree depth

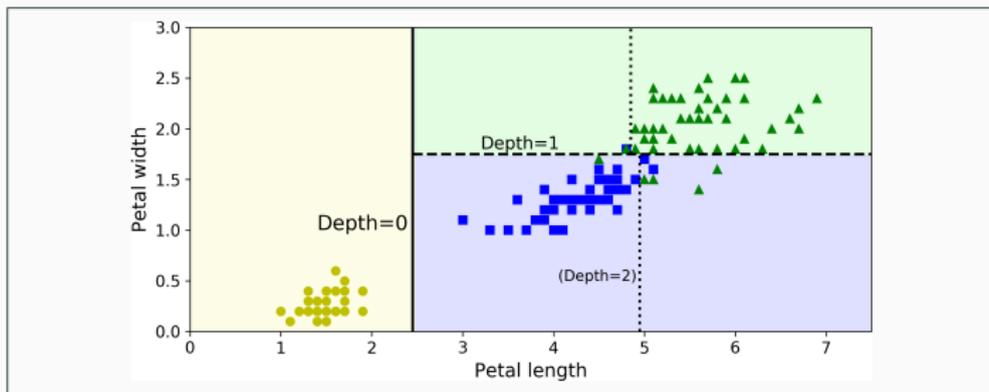


Figure 6-2. Decision Tree decision boundaries

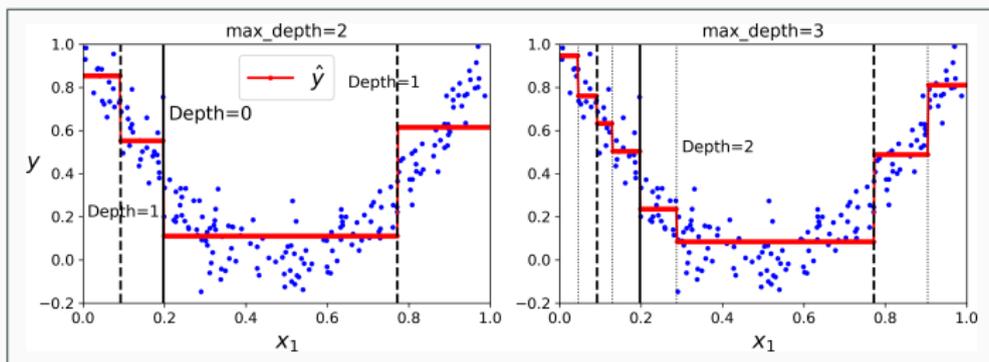


Figure 6-5. Predictions of two Decision Tree regression models

# Training algorithm for decision trees

## CART (Classification and regression tree)

- Leo Breiman (1984), 분류와 회귀 모두 학습 가능
- Scikit-Learn에서는 CART만 지원
- C4.5와 일정 부분 유사하지만 CART는 numerical 변수도 지원, 입력 자료를 나누는 과정 등에서 차이

## ID3 (Iterative Dichotomiser 3)

- Ross Quinlan (1986), CART와 비슷한 시기에 독립적으로 발명
- CART와 비슷한 결과
- Multiway tree 지원

C4.5 (ID3  $\rightarrow$  C4.5  $\rightarrow$  C5.0): Numerical 변수도 지원

그 밖에 CHAID, MARS, QUEST, Conditional inference tree 등이 있음

# CART training algorithm

## Tree growing algorithm

- 학습 데이터셋을 어떤 feature  $k$ 에 대해 임계값(thresholding)  $t_k$ 로 분리
- 부분 집합들을 같은 방법으로 분리해 나감
- Stopping condition에 도달할 때까지 계속 분리
- Stopping condition (hyperparameters):  
max\_depth, min\_sample\_split, in\_samples\_leaf, min\_weight\_fraction\_leaf, max\_leaf\_nodes 등

## Greedy CART

- CART 알고리즘은 greedy algorithm
- 각 단계에서 최적의 해를 찾으며 다음 단계로 진행하지만, 지금의 선택이 몇 단계가 지나서까지 최적인지 판단하지 않음
- 최적 해를 보장하지 않음

## NP-complete problem

- 최적의 binary decision tree를 구성하는 문제는 NP-complete
- 매우 작은 문제가 아니라면 intractable임  $\implies$  상당히 좋은 해로 만족할 수 밖에 없음

# Computational complexity

- $n$ : 노드 수,  $h$ : 트리의 깊이,  $N$ : 샘플 수
- Binary tree의 예측 복잡도 (prediction complexity)
  - 트리의 깊이 만큼만 비교 판단하면 됨
  - 깊이  $h$ 인 tree의 최대 노드 수  $n$ 은

$$n = 1 + 2 + \dots + 2^{h-1} + 2^h = 2^{h+1} - 1 \quad (1)$$

양변에  $\log_2$ 를 취하면

$$h = \log_2(n + 1) - 1 \quad (2)$$

그러므로 계산복잡도  $O(\log_2 n) \implies$  feature 수와 무관

- Binary tree의 학습 복잡도 (training complexity)
  - $N$ 개의 instance가 모두 최종 leaf까지 ( $h$ 단계) 가야 할 수 있으므로  
계산 복잡도  $N \times h = O(N \log_2 n)$

# Gini impurity measure

## Splitting Criteria

$$\arg \max_k J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}} \quad (3)$$

- $G_{\text{left or right}}$ : left/right의 impurity
- $m_{\text{left or right}}$ : left/right 부분 집합의 instance 수

## Gini impurity measure

- 임의의 node  $i$ 에서 impurity  $G_i$

$$G_i = 1 - \sum_k p_{i,k}^2 \quad (4)$$

- $p_{i,k}$ : node  $i$ 에서 클래스  $k$ 에 속하는 instance의 비율
- e.g.,

$$G = 1 - \left(\frac{49}{54}\right)^2 - \left(\frac{5}{54}\right)^2 \approx 0.168 \quad (5)$$

gini = 0.168  
samples = 54  
value = [0, 49, 5]  
class = versicolor

# Information gain measure

## Splitting Criteria

$$\arg \max_k IG(k, t_k) = H_{\text{before split}} - H_{\text{after split}} \quad (6)$$

## Information gain (entropy)

- 한 노드에서의 엔트로피

$$H = - \sum_k p_{i,k} \log p_{i,k} \quad (7)$$

- $p_{i,k}$ : node  $i$ 에서 클래스  $k$ 에 속하는 instance의 비율
- e.g.,

$$H = -\frac{49}{54} \log \frac{49}{54} - \frac{5}{54} \log \frac{5}{54} \approx 0.31 \quad (8)$$

gini = 0.168  
samples = 54  
value = [0, 49, 5]  
class = versicolor

## Decision trees와 과적합(overfitting)

- 의사 결정 트리는 데이터에 대한 가정이 별로 없음
- Tree growing에 제약을 주지 않으면 과적합되기 쉬움

## 과적합 방지를 위한 regularization

- Tree growing에 있어 자유도에 제한을 가하는 것
- Regularization 방법은 알고리즘에 따라 다름

# Regularization hyperparameters

Hyperparameter	Default	Description
max_depth	none	트리의 최대 깊이
min_samples_split	2	내부 node를 split하기 위한 최소 샘플 수
min_samples_leaf	1	leaf node가 되기 위한 node 내 샘플 수
min_weight_fraction_leaf	0	min_samples_leaf와 같으나 샘플 수 비율로 표시
max_leaf_nodes	none	leaf node 최대 수
max_features	none	각 node에서 split을 위해 계산하는 최대 feature 수

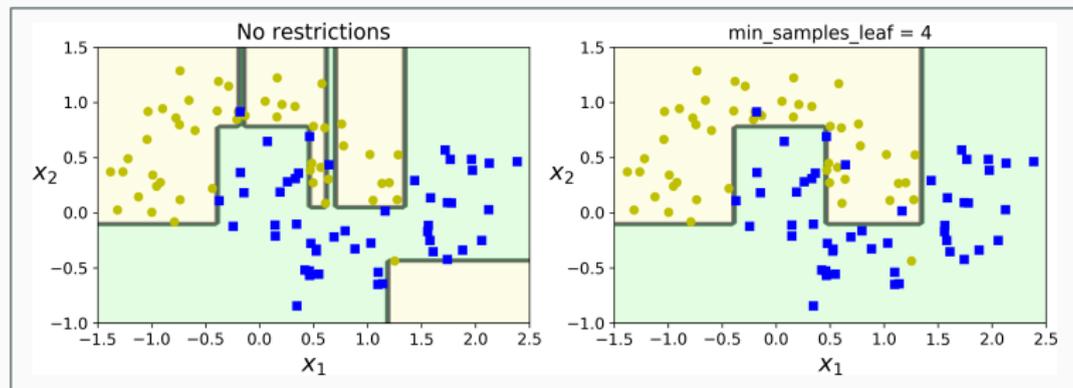


Figure 6-3. Regularization using `min_samples_leaf`

# Instability of decision tree

Decision boundary가 항상 좌표축에 수직  $\implies$  데이터셋의 회전에 민감

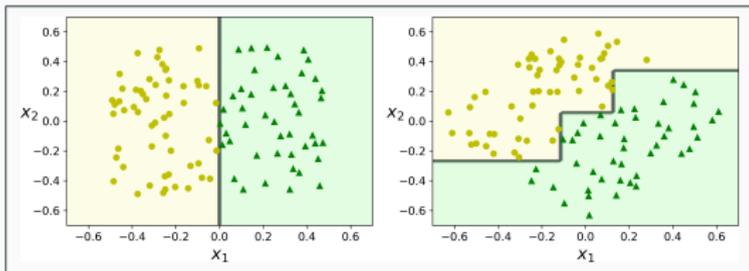
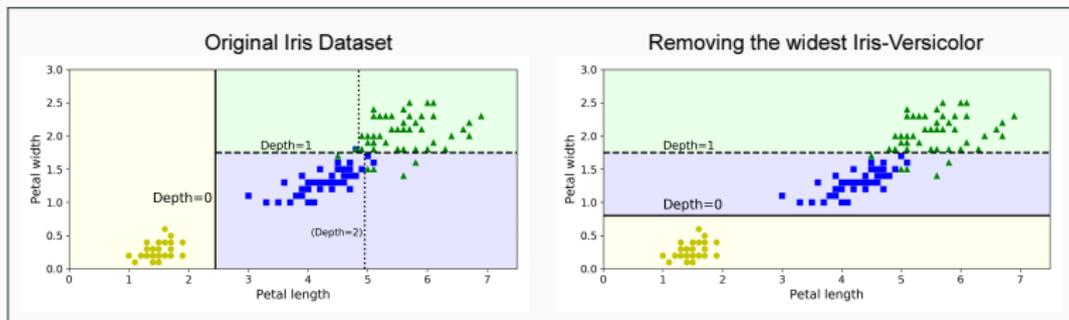


Figure 6-7. Sensitivity to training set rotation

데이터의 작은 변화에도 매우 민감



# Ensemble Learning

---

## 앙상블 러닝이란

- 한 전문가의 의견보다 여러 사람의 종합된 의견이 더 나은 경우가 많음
- 하나의 좋은 예측기(predictor, classifier or regressor)보다 보통 예측기 집단의 예측이 더 나옴
- 결과가 중요한 의사 결정에 직면하면, 결정을 도와줄 다른 전문가의 의견을 구하게 되어 있음

## 앙상블 러닝의 목표

- 여러 다양한 의견을 깊이 고려
- 논리적 과정을 통해 결합
- 결정에 대한 신뢰성을 높임

## 뛰어난 성과

- Netflix Competition, KDD 2009, Kaggle 같은 세계적인 머신 러닝 대회에서 많은 우승 팀들이 앙상블 러닝을 사용
- Competition 참가자, 수상자들이 가장 선호하는 머신 러닝 알고리즘

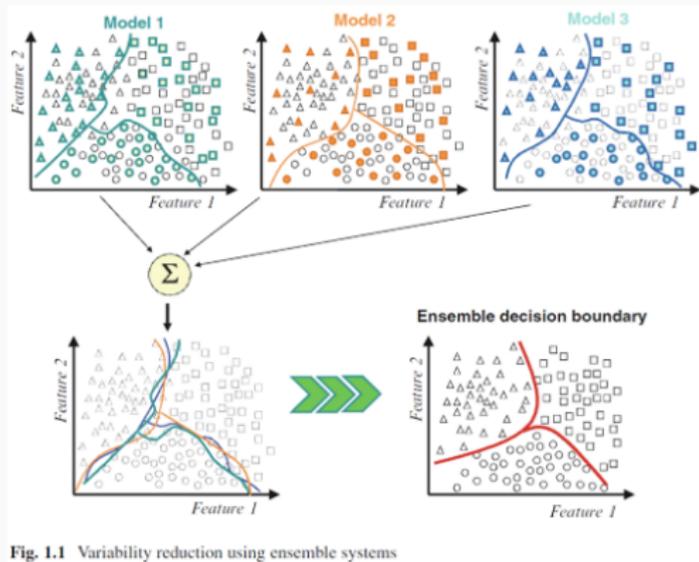
## 앙상블 방법

- Bagging
- Boosting
- Stacking

# Ensemble learning

## 앙상블을 통한 변동성 축소

- 분류기들의 오류가 각 샘플에 대해 다른 오류를 발생시키지만 옳은 분류에 대해서는 일반적으로 일치한다고 가정
- 앙상블의 분류기 출력을 averaging 하는 것은 오류 요소들이 averaging out 되게 만들어 앙상블 모델 오류를 감소



# Voting Classifiers

## Hard voting classifier

- 여러 개의 classifier들을 사용하여 다수결에 따른 예측을 하는 분류기
- 앙상블 내의 가장 우수한 분류기보다 좋은 결과를 내기도 함

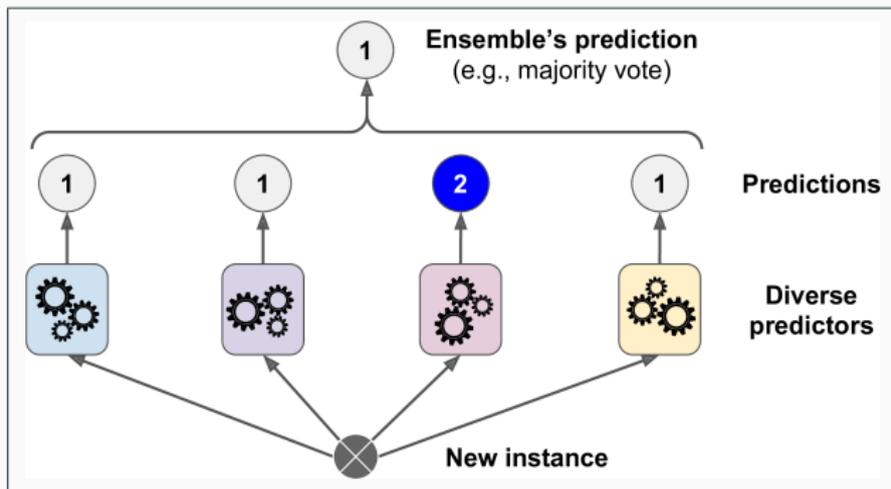


Figure 7-2. Hard voting classifier predictions

## Weak learner & strong learner

- 각 분류기가 모두 weak learner (or classifier)이더라도 앙상블은 strong learner가 될 수 있음
- Weak learner: 정확도가 50%보다 약간 좋은 정도
- Strong learner: 정확도가 상당히 높은 분류기 (70% - 80% 이상, 주관적)

## Ensemble and a law of large numbers

- 정확도가 51%(오류율 49%)인 weak learner들로 된 앙상블이 어떻게 높은 정확도를 낼 수 있는가?
- 앞면이 나올 확률이 51%인 동전을 1,000번 던질 때, 앞면이 과반일 확률  $\approx 75\%$
- 정확도가 51%인 weak learner 1,000개로 구성된 앙상블의 과반 득표 클래스 예측 정확도  $\approx 75\%$

# Voting Classifiers

A (strong) law of large numbers (큰 수의 법칙, 라플라스의 정리)

$$P\left(\lim_{n \rightarrow \infty} \frac{1}{N} \sum_k X_k = \mu\right) = 1 \quad (9)$$

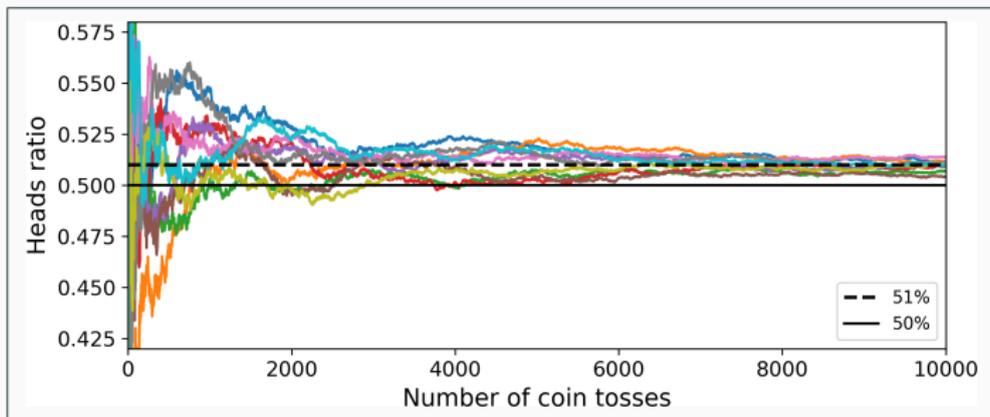


Figure 7-3. The law of large numbers

## Soft voting classifier

- 모든 classifier가 클래스 확률을 예측할 수 있는 경우
- 즉, `predict_probability()` 등의 method가 있는 경우
- `sklearn.ensemble.VotingClassifier()` : `voting = 'soft'`
- 모든 classifier들이 출력하는 클래스 확률값들의 평균에서 값이 최대인 클래스라고 예측
- 신뢰도가 높은 classifier에게 더 큰 가중치를 주게 되므로 hard voting보다 일반적으로 성능이 더 좋음

## Diversity(다양성)의 확보

- 여러가지 다른 독립적 base classifier들을 사용 (hard voting or soft voting classifiers)
- 훈련 데이터의 서로 다른 부분집합들을 사용  $\implies$  가장 흔한 방법
- 서로 다른 샘플링 방법들을 사용
  - Bootstrapping  $\implies$  bagging
  - 이전에 오분류된 샘플들을 선호하는 분포에서 샘플링  $\implies$  boosting
- 가용한 feature들의 서로 다른 부분집합을 사용  $\implies$  random subspace method
- Base classifier에 서로 다른 파라미터들을 적용
- 여러가지 다른 형태의 diversity 측도가 있음
- 그러나, diversity와 ensemble accuracy간의 구체적인 관계는 아직 밝혀지지 않음

# Bagging and pasting

훈련 세트의 다른 부분집합들을 사용하여 다양성을 확보

- Bagging (Bootstrap AGGregatING): sampling with replacement
- Pasting: sampling without replacement
- 모든 predictor의 훈련이 끝나면, 앙상블은 새로운 사례에 대해서 모든 predictor들의 예측을 aggregating (수집, 종합)해서 예측을 함

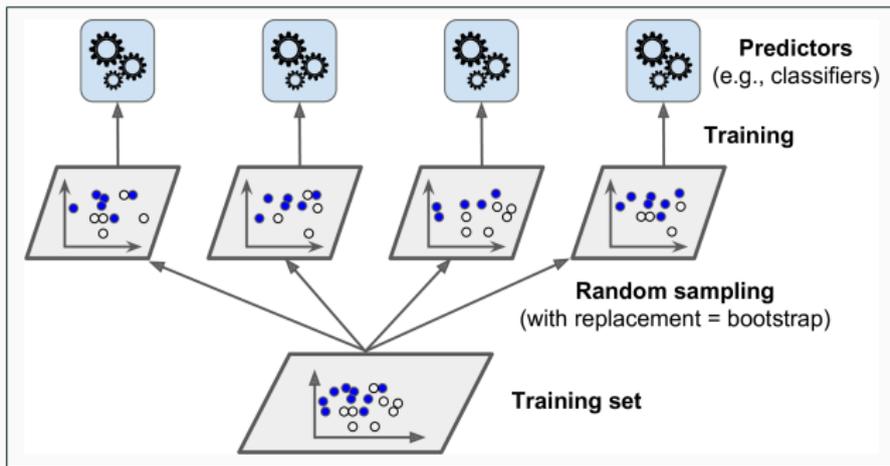


Figure 7-4. Pasting/bagging training set sampling and training

- Aggregation function
  - Classification의 경우  $\implies$  Statistical mode (최빈값, 다수결 등)
  - Regression의 경우  $\implies$  Average
- 개별 predictor는 원래 훈련 데이터셋에서 학습한 것보다 bias가 높음
- Aggeragation을 통해 bias와 variance를 모두 줄일 수 있음
- Bagging과 Pasing은 각 predictor를 병렬로 학습시킬 수 있음 (CPU, GPU)  
 $\implies$  확장성이 좋다

# Bagging and pasting

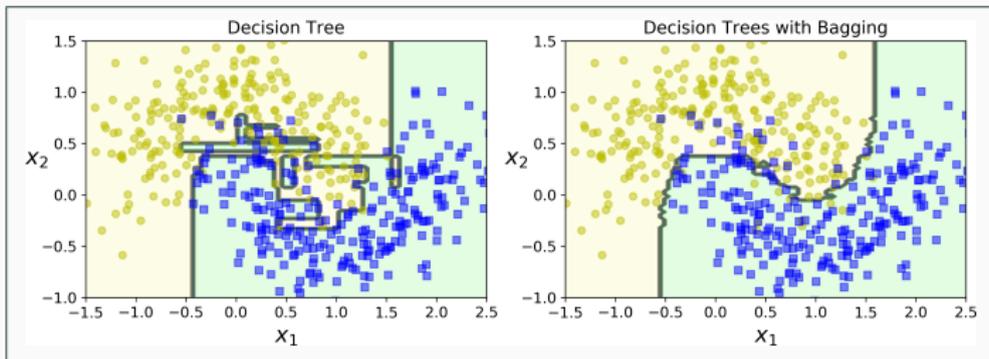


Figure 7-5. A single Decision Tree versus a bagging ensemble of 500 trees

- Bootstrapping은 각 predictor가 훈련하는 부분집합의 다양성을 증가시키므로 bagging이 pasting보다 bias가 약간 더 높아짐 (predictor들 사이의 상관관계를 줄여줌)  $\implies$  앙상블의 variance를 줄여줌
- 일반적으로 pasting보다 bagging의 결과가 더 좋음
- 데이터, 시간, 컴퓨팅 파워에 여유가 있다면 bagging과 pasting의 결과를 비교 평가하기를 권장

# Out-of-bag evaluation

## Bootstrapping의 특징

- 전체 데이터셋에서 평균적으로 63%의 사례만 추출되고 나머지 37%는 훈련에 사용되지 않고 남아있게 됨
- $m$ 개의 샘플 중에서 무작위로 하나를 추출할 때 선택되지 않고 남아있을 확률은  $(1 - \frac{1}{m})$
- 따라서  $m$ 번 추출하여도 선택되지 않고 남아있을 확률은  $(1 - \frac{1}{m})^m \approx \frac{1}{e} \approx 0.36788 \implies$  Out-of-bag (OOB) instances

별도의 validation set 또는 cross validation 없이 OOB 사례들로 앙상블을 평가할 수 있음

- 앙상블의 evaluation score는 각 predictor의 OOB evaluation 결과의 평균
- BaggingClassifier에서 oob\_score = True
- oob\_score\_: evaluation 결과 점수
- oob\_decision\_function\_: 훈련 사례의 클래스 확률을 보여주는 변수

# Appendix

---

## Reference and further reading

- “Chap 6 | Decision Trees” and “Chap 7 | Ensemble Learning and Random Forest” of A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow
- “Lecture 5 | Decision Trees” and “Lecture 7 | Ensemble Learning and Random Forest” of Kwang Il Kim, Machine Learning (2019)