머신러닝 교과서

3장. 사이킷런을 타고 떠나는 머신러닝 분류 모델 투어

2020. 1. 13(Mon)
SK주식회사 C&C
강천성

Logistic Regression

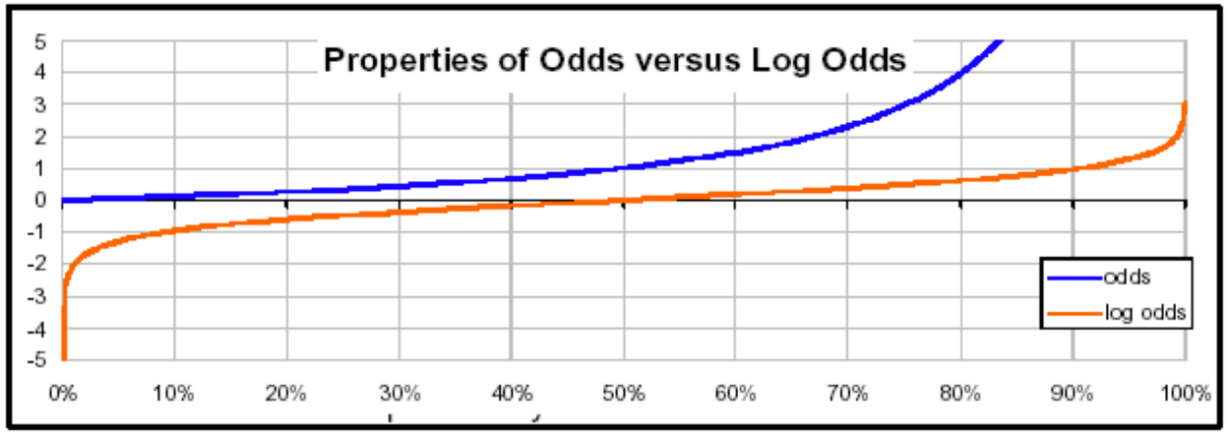
Regression 이라는 말에서 알 수 있듯이, 로지스틱 회귀 모델은 선형 회귀 모델에서 변형된 모델입니다.
Odds라는 어떤 일이 발생할 상대적인 비율 개념을 사용해 선형 회귀식을 변형합니다.

$$Odds = \frac{p}{1-p}$$

p : 어떤 일이 발생할 확률

Odds를 그대로 사용하지말고 log를 취해 사용하면 0을 기준으로 상호 대칭적이며, 계산을 수월하게 할 수 있도록 변경해줍니다.
기존의 선형 회귀식에서 y 위치에 log Odds를 적용하면 다음과 같은 식이 됩니다.

$$ln(\frac{Y}{1-Y}) = wx + b$$



이를 y에 대해 정리하면 그 유명한 sigmoid 식이 됩니다.

$$y = \frac{1}{1+\exp^{-(wx+b)}}$$

Logistic Regression

Linear Regression은 잔차의 제곱을 최소화 하는 방식으로 학습을 했었습니다.
Logistic Regression은 Maximum Likelihood Estimation(MLE)이라는 과정을 통해 모델을 학습하는데, 자세한 내용은 참조 목록에 있는 페이지를 확인해주시면 감사하겠습니다.

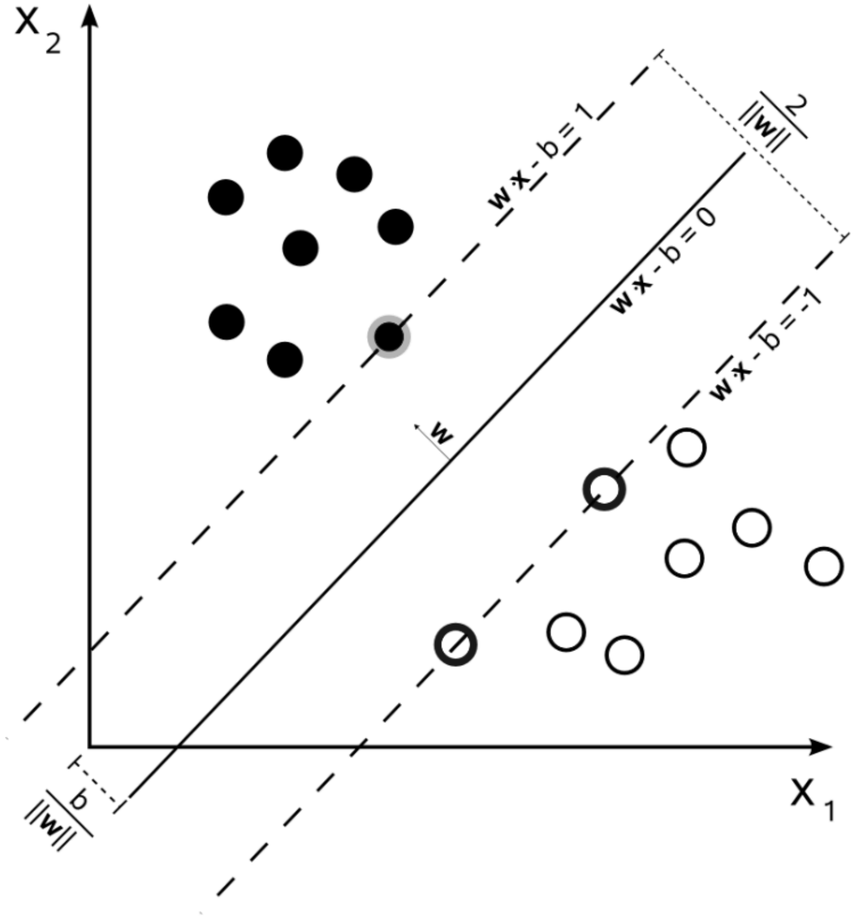
로지스틱 회귀은 이진 분류 모델로 알고 있는데, 어떻게 여러개의 클래스를 분류할 수 있나요?

하나의 수식이 출력하는 결과는 클래스의 확률을 나타내는 것은 맞습니다. 하지만, 멀티 클래스인 경우 내부적으로 클래스 수에 맞게 여러개의 수식을 만들어 각각의 클래스에 속할 확률을 계산한 후 가장 높은 확률은 가진 클래스로 분류합니다. 이를 One-vs-Rest라고 합니다. 자세한 내용은 참조 목록에 있는 페이지를 확인해주시면 감사하겠습니다.

- **Maximum Likelihood Estimation, 최대 우도 추정 : https://ratsgo.github.io/statistics/2017/09/23/MLE/**
- **One-vs-Rest : https://datascienceschool.net/view-notebook/7a6b958e9d51451689138cca93a047d8/**

Support Vector Machine

Support Vector Machine(SVM, 서포트 벡터 머신)는 주어진 데이터를 바탕으로하여 두 카테고리(이진 분류의 경우) 사이의 간격(Margin, 마진)을 최대화하는 데이터 포인트(Support Vector, 서포트 벡터)를 찾아내고,
그 서포트 벡터에 수직인 경계를 통해 데이터를 분류하는 알고리즘입니다.



Support Vector Machine

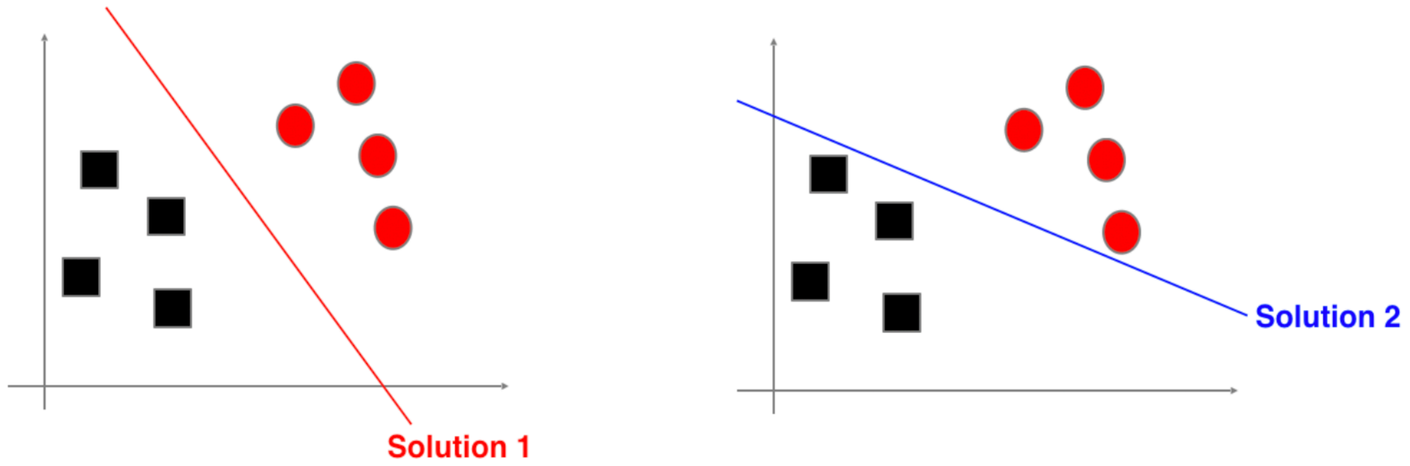
왜 마진을 최대화 할까요?

서포트 벡터 머신에서 나오는 마진은 물건을 판매할때 마진이 20%다 라고 말하는 그 마진이 맞습니다.
그렇다면 경계면과의 마진을 최대화 하는 것이 왜 분류를 잘하게 할까요?

경험적 위험 최소화(Empirical Risk Minimization, ERM) vs 구조적 위험 최소화(Structural Risk Minimization,SRM)

- 경험적 위험 최소화
  - 훈련 데이터에 대해 위험을 최소화
  - 학습 알고리즘의 목표
  - 뉴럴 네트워크, 결정 트리, 선형 회귀, 로지스틱 회귀 등.
- 구조적 위험 최소화
  - 관찰하지 않은(Unseen) 데이터에 대해서도 위험을 최소화
  - 오차 최소화를 일반화 시키는 것

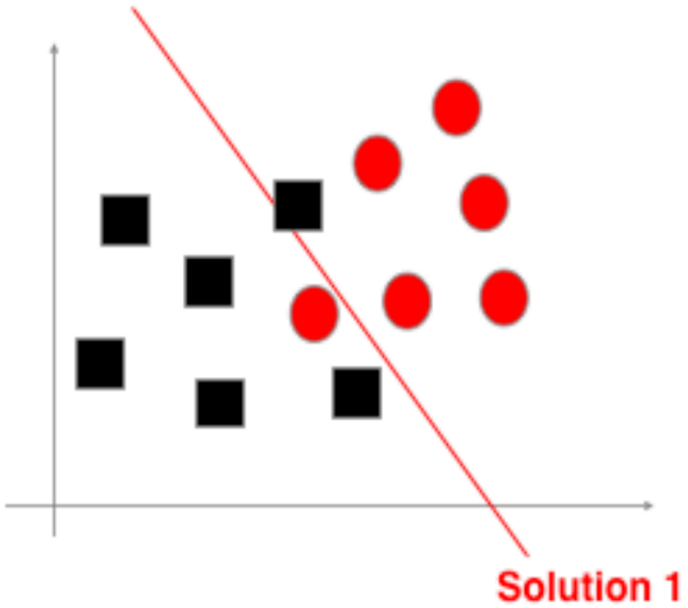
어떤 모델이 더 좋을까요?

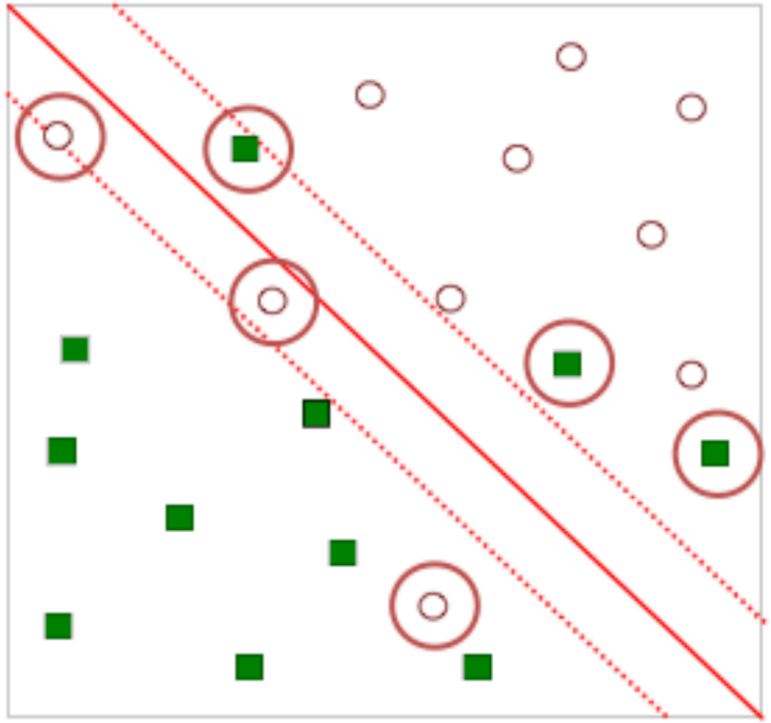


Support Vector Machine

Cost : Soft or Hard

SVM에는 Soft Margin, Hard Margin 이라는 말이 있습니다. 단어 자체에서도 유추할 수 있으시겠지만, Soft Margin은 유연한 경계면을 만들어내고 Hard Margin은 분명하게 나누는 경계면을 만들어냅니다.
그렇다면 왜 Soft Margin이 필요한걸까요?

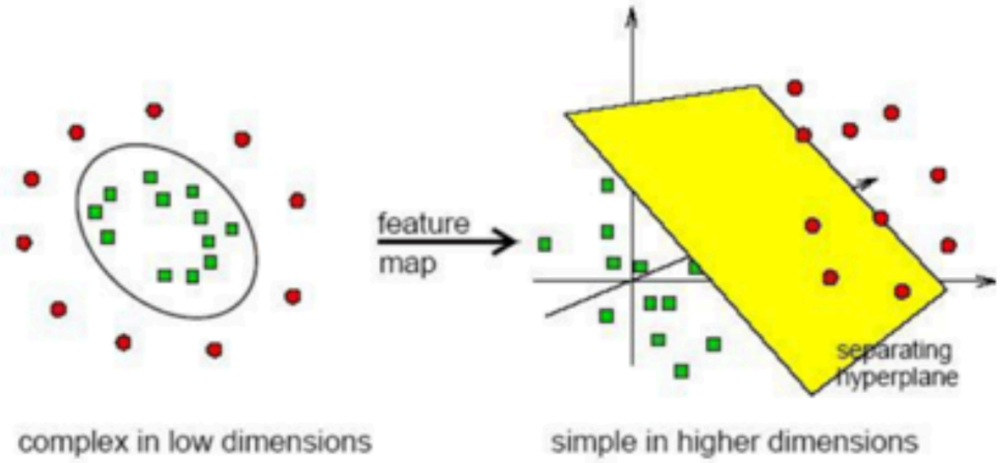




Support Vector Machine

저차원을 고차원으로 Kernel Trick

SVM은 기본적으로 선형 분류를 위한 경계면을 만들어냅니다. 그렇다면 어떻게 비선형 분류를 할 수 있을까요?



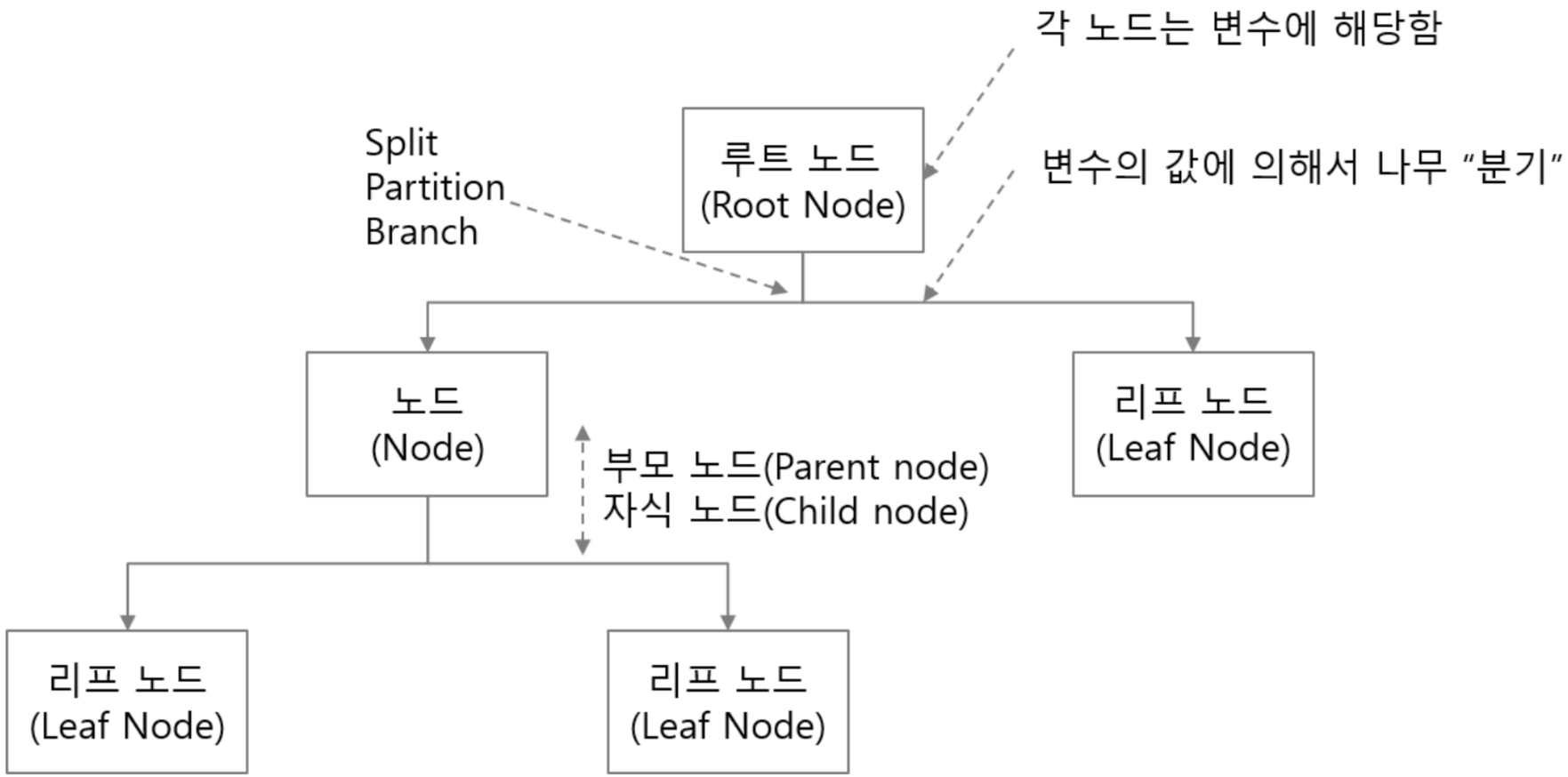
대표적인 Kernel 함수

- Linear (선형 함수)
- Poly (다항식 함수)
- RBF (방사기저 함수)
- Hyper-Tangent (쌍곡선 탄젠트 함수)

저차원(2차원)에서는 선형 분리가 되지 않을 수 있지만, 고차원(3차원)에서는 선형 분리가 가능할 수 있습니다.
이러한 원리를 바탕으로 선형 분리가 불가능한 저차원 데이터를 선형 분리가 가능한 어떤 고차원으로 보내 선형 분리를 할 수 있습니다.
하지만, 저차원 데이터를 고차원으로 보내서 서포트 벡터를 구하고 저차원으로 내리는 과정에서 더 복잡해지고 연산량도 많아질것이 분명합니다.
그래서 여기에서 Kernel Trick이라는 Mapping 함수를 사용합니다. Kernel Trick은 고차원 Mapping과 고차원에서의 내적 연산을 한번에 할 수 있는 방법입니다.
이를 통해 여러가지 Kernel 함수를 통해 저차원에서 해결하지 못한 선형 분리를 고차원에서 해결할 수 있습니다.

Decision Tree

결정 트리는 입력 변수를 특정한 기준으로 잘라(분기) 트리 형태의 구조로 분류를 하는 모델입니다.



Decision Tree

- 사람의 논리적 사고 방식을 모사하는 분류 방법론
- IF-THEN rule의 조합으로 class 분류
- 결과를 나무 모양으로 그릴 수 있음
- Greedy 한 알고리즘 (한번 분기하면 이후에 최적의 트리 형태가 발견되더라도 되돌리지 않음, 최적의 트리 생성을 보장하지 않음)
- 축에 직교하는 분기점
- 데이터 전처리가 필요 없음

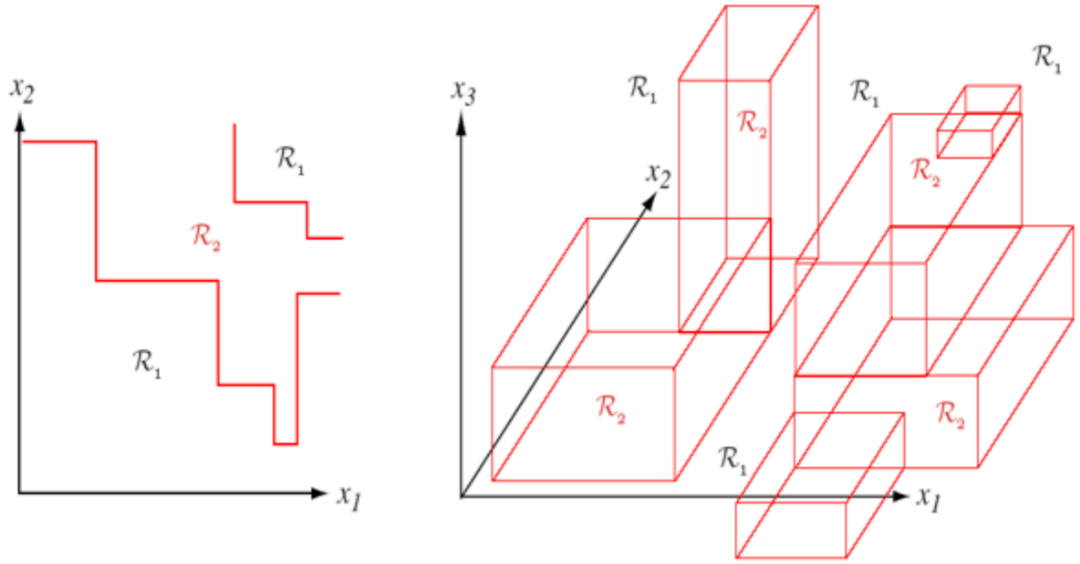
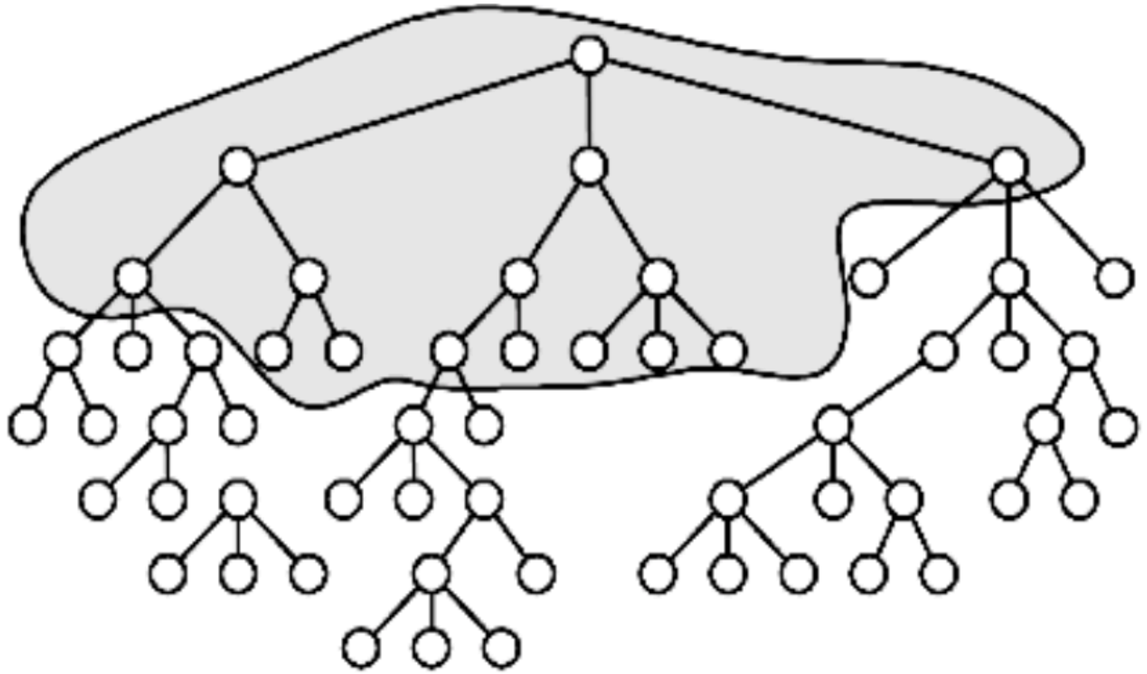


FIGURE 8.3. Monothetic decision trees create decision boundaries with portions perpendicular to the feature axes. The decision regions are marked $\mathcal{R}_1$ and $\mathcal{R}_2$ in these two-dimensional and three-dimensional two-category examples. With a sufficiently large tree, any decision boundary can be approximated arbitrarily well in this way. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Decision Tree

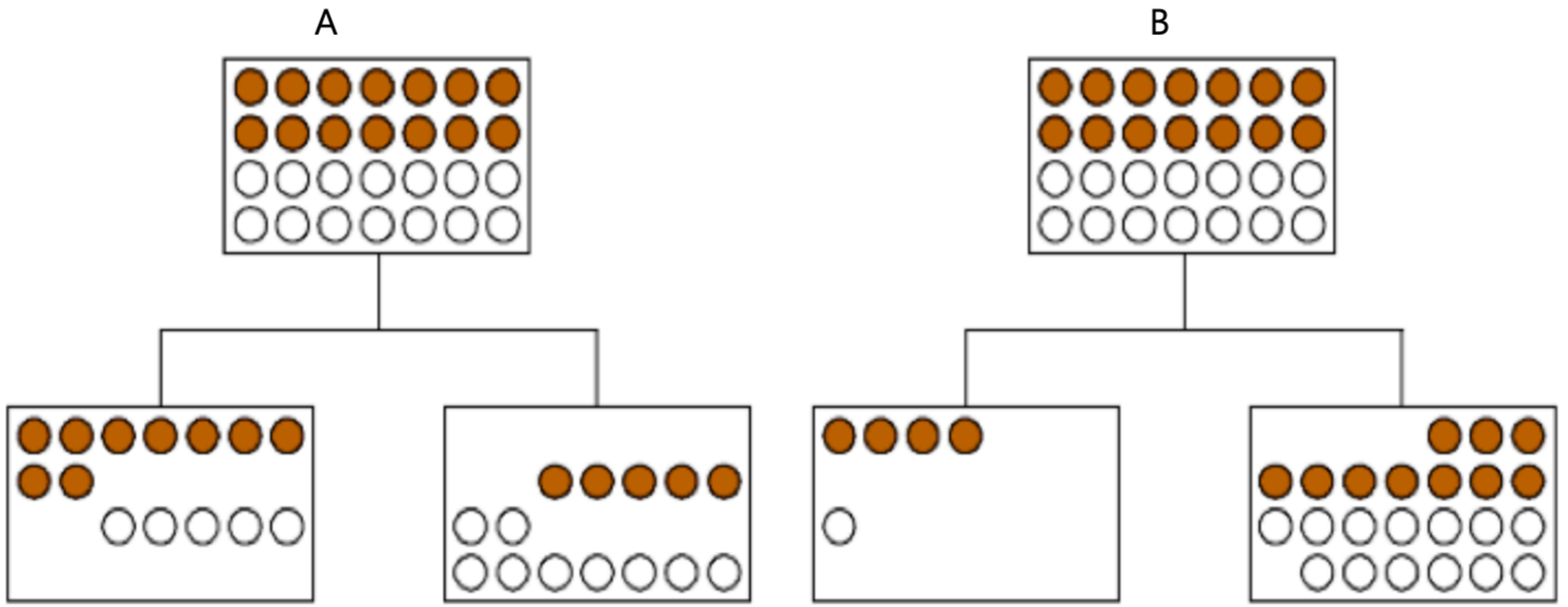
가지치기 (Prunning)
깊은 Decision Tree는 학습 데이터에 과적합 되는 경향이 있으므로
가지치기를 통해 일반화 성능을 올립니다.



Decision Tree

불순도(Impurity), ex) Gini, Entropy

결정 트리는 데이터의 불순도를 최소화 할 수 있는 방향으로 트리를 분기합니다.
불순도란 정보 이론(Information Theory)에서 말하는 얻을 수 있는 정보량이 많은 정도를 뜻합니다.
ex) 오늘 해가 동쪽에서 뜰꺼야 -> 낮은 정보량, 오늘 일식이 일어날꺼야 -> 높은 정보량

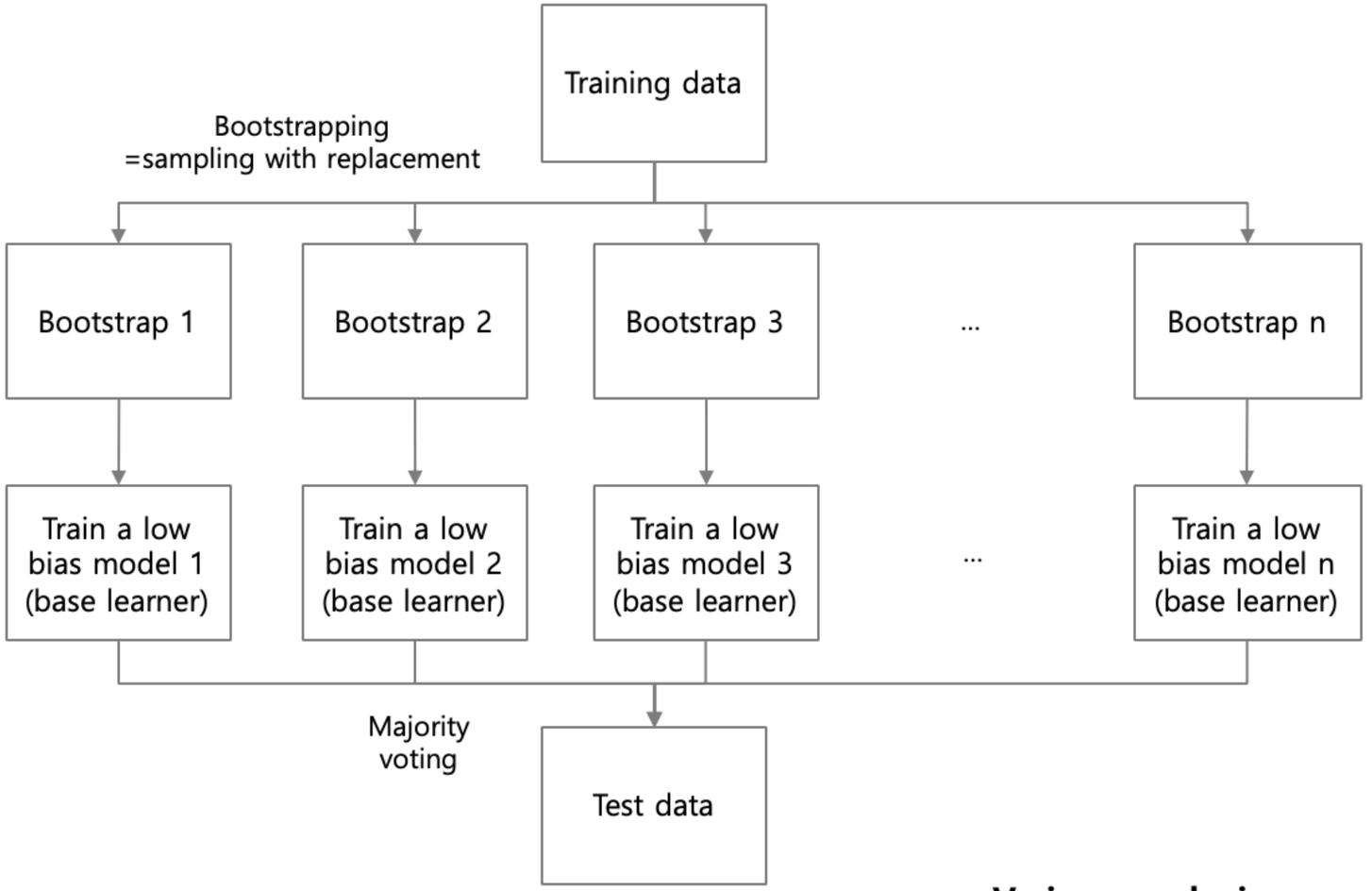


Entropy = 0.9403
(Gini index = 0.541)

Entropy = 0.9402
(Gini index= 0.539)

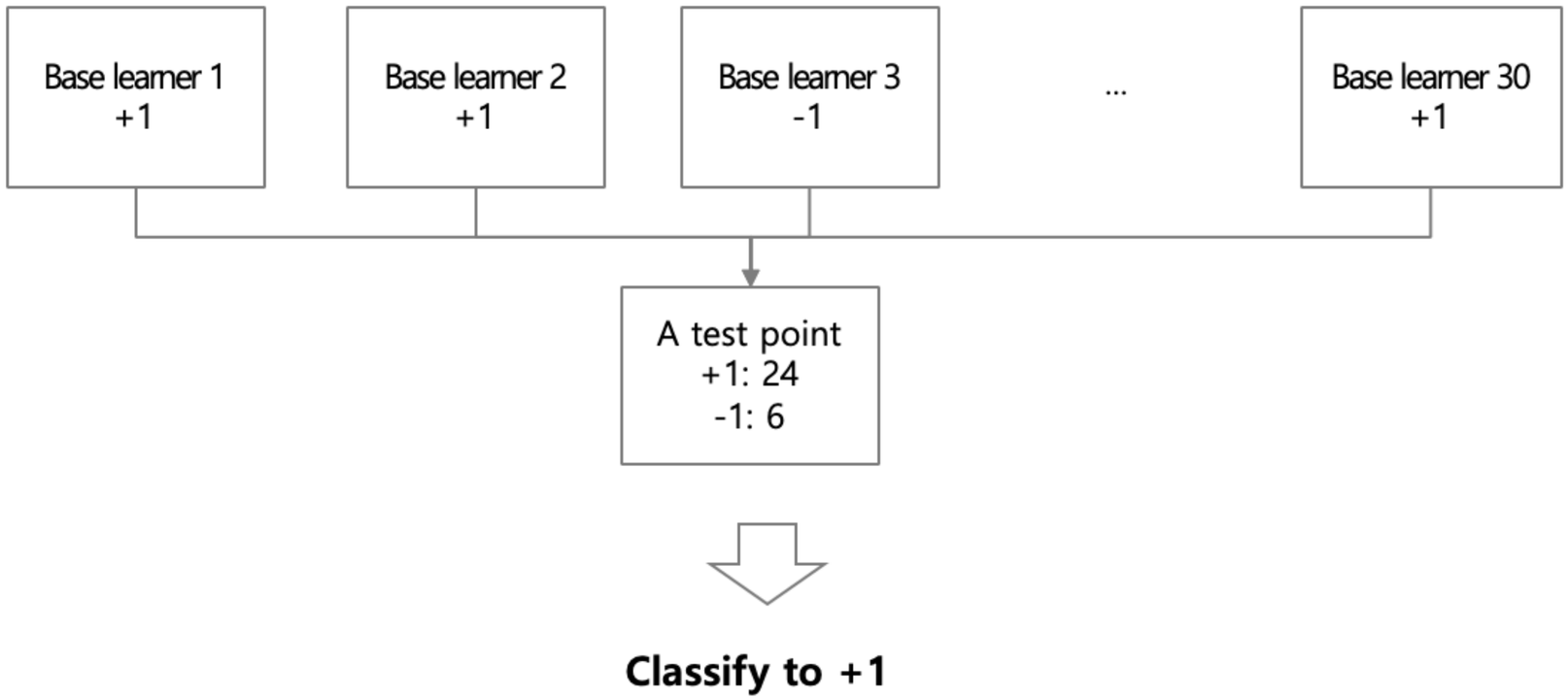
Random Forest

앙상블: 배깅(Bagging)



Variance reducing
with low bias & high variance base learners

Random Forest



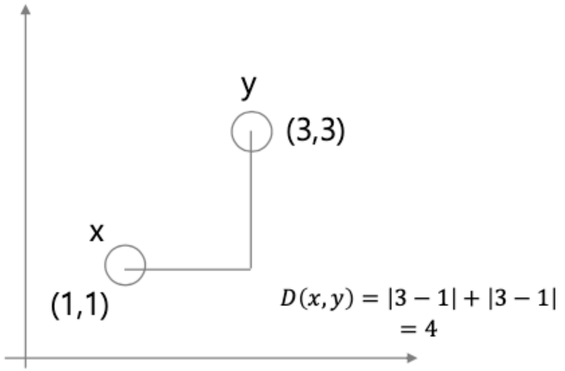
k-Nearest Neighbor

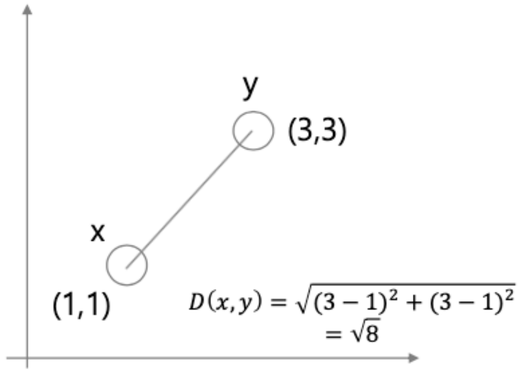
- 먼저 뿌려진 데이터 중 k 개의 이웃을 선택해서 투표하는 방식의 분류 알고리즘이다.
- 따로 학습 단계를 수행하지 않는다.
- 판별 함수를 학습하는 것이 아니라 훈련 데이터셋을 메모리에 저장하는 게으른 학습기(Lazy Learner)이다.
- 데이터를 학습하지 않기 때문에 새로운 훈련 데이터에 즉시 적용할 수 있는 장점이 있다.
- 거리 기반 알고리즘이다.

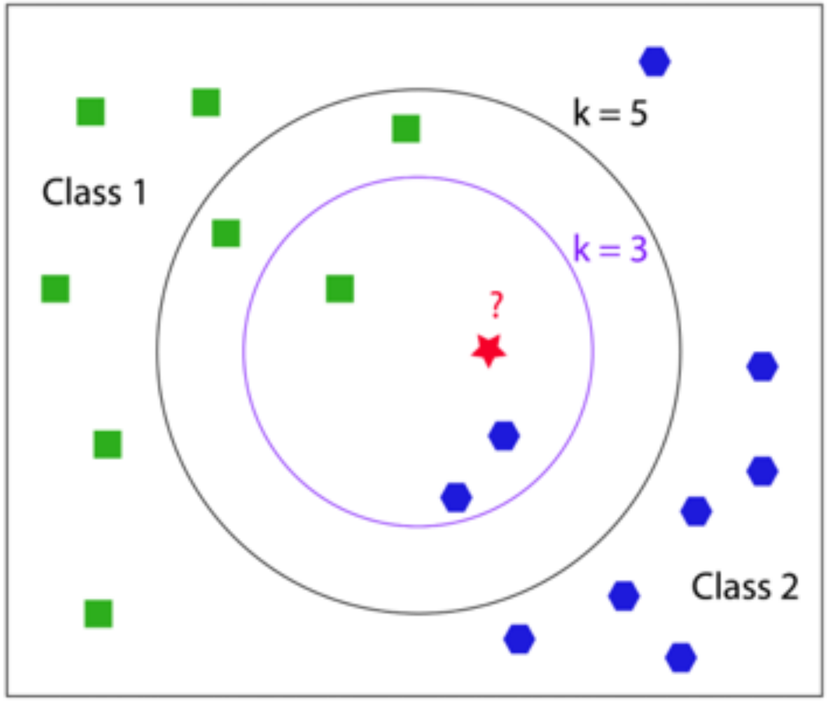
Manhattan distance vs. Euclidean distance

$$D(x,y) = \sum_{i=1}^{d} |x_i - y_i|$$

$$D(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$







Scikit-Learn 문법의 공통점

1. 모델 불러오기 및 정의

```
from sklearn.svm import SVC
clf = SVC(C=1.0, kernel='rbf', random_state=2019)
```

- hyper-parameter 세팅

2. fit

```
clf.fit(x_train, y_train)
```

- (훈련)데이터로 모델 학습 또는 특징 추출

3. predict or predict_proba or transform

```
y_pred = clf.predict(x_test)
```

- (테스트)데이터 라벨(확률) 예측 또는 변환

4. scoring

```
accuracy_score(y_test, y_pred)
```

- 정확도, AUC, R^2 등 적절한 스코어 함수로 결과 확인