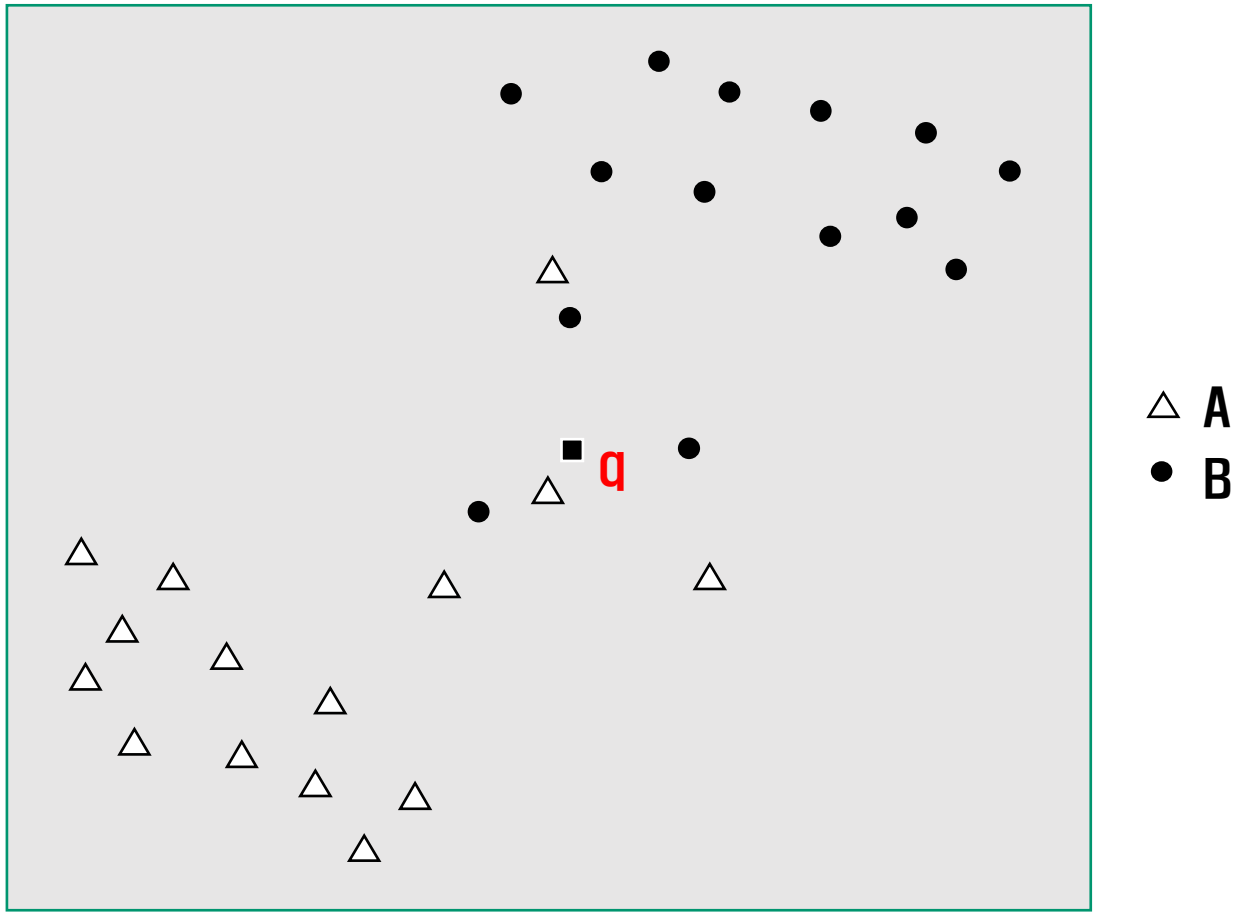
kNN

kNN (k-Nearest Neighbor)개요

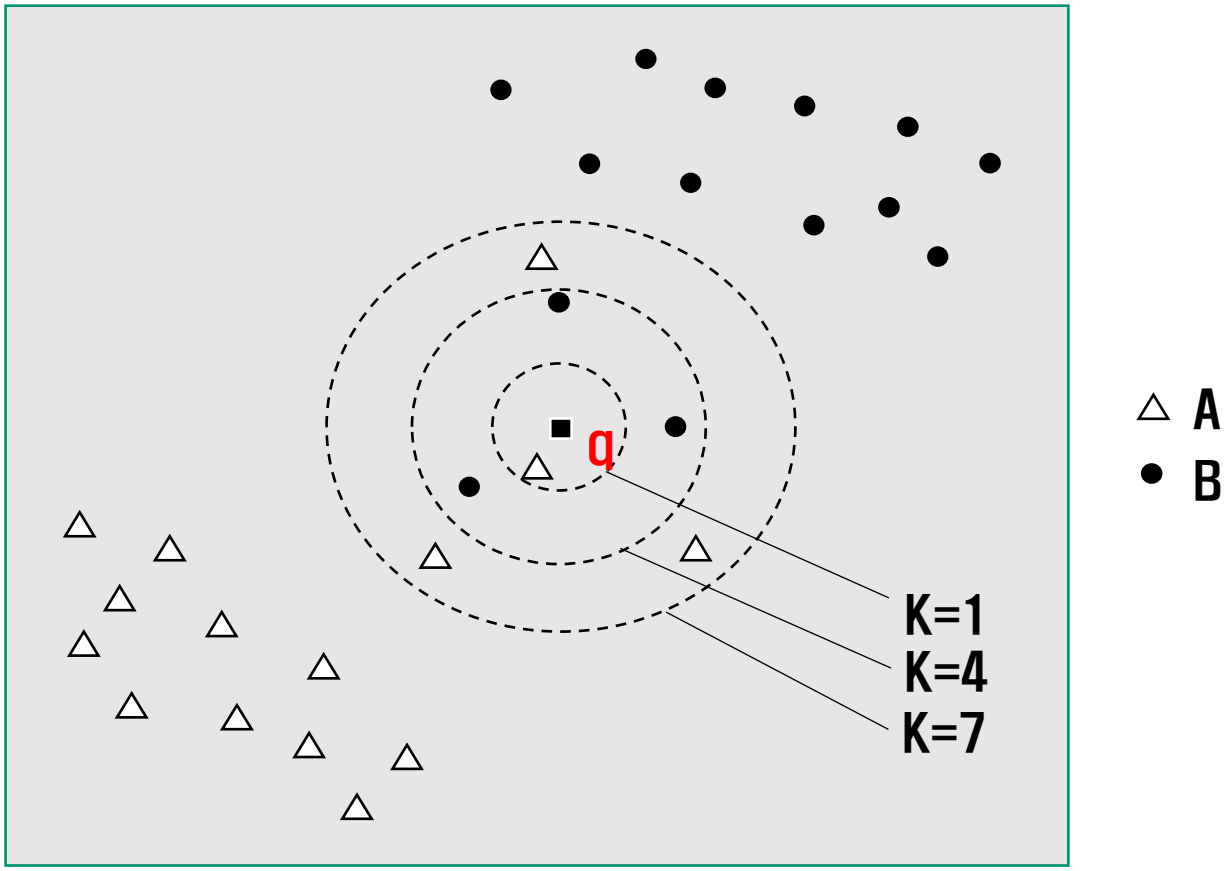
- kNN 알고리즘
  - 주어진 샘플의 특성 값과 가장 가까운 특성을 가진 이웃(neighbor)을 k개 선택
  - 선택된 이웃들 레이블의 평균치로 이 샘플이 속할 category를 예측
- 직관적으로 이해하기 쉬운 분류 알고리즘으로 추천 시스템에서 많이 사용
  - 적절한 추천을 하기 위해서 추천을 요청한 사람의 성향을 특성들로 파악
  - 그 사람과 가장 성향이 유사한 k명의 사람들이 좋아하는 품목을 추천
- 협업 필터링(collaborative filtering)
  - 서점에서 도서 추천
    - 먼저 추천 고객 A와 독서 취향이 비슷한 사람들의 그룹을 찾고
    - 그 그룹에서 평이 가장 좋은 책들 중에서 고객 A가 아직 보지 않은 책을 추천

    (서점은 평소 고객들이 어떤 책을 즐겨 보는지, 책에 대한 평가는 어떠했는지 등 조사
    → 비슷한 취향을 구분)
  - 사용자 기반 협업 필터링(User-based collaborative filtering)
  - 아이템 기반 협업 필터링(Item-based collaborative filtering)
    - a 라는 상품을 좋아한 사람들은 대부분 b나 c 상품도 같이 좋아한다는 사실을 이용
    - 상품을 기준으로 유사한 상품들을 Grouping
    - 어떤 사람이 새로운 상품의 추천을 원할 경우, 유사한 상품 그룹에서 추천

kNN 동작



kNN 동작

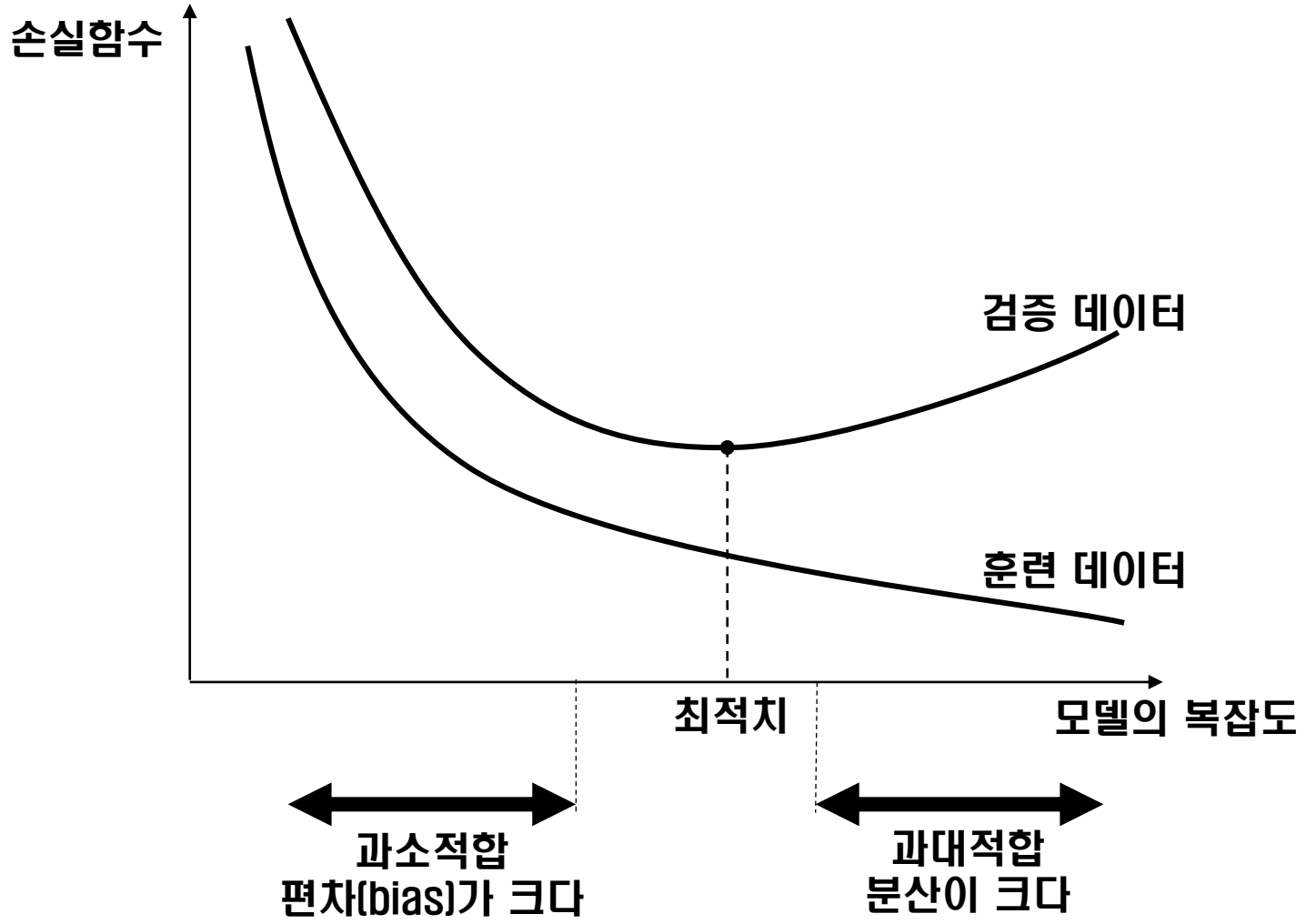


k 값에 따른 영향

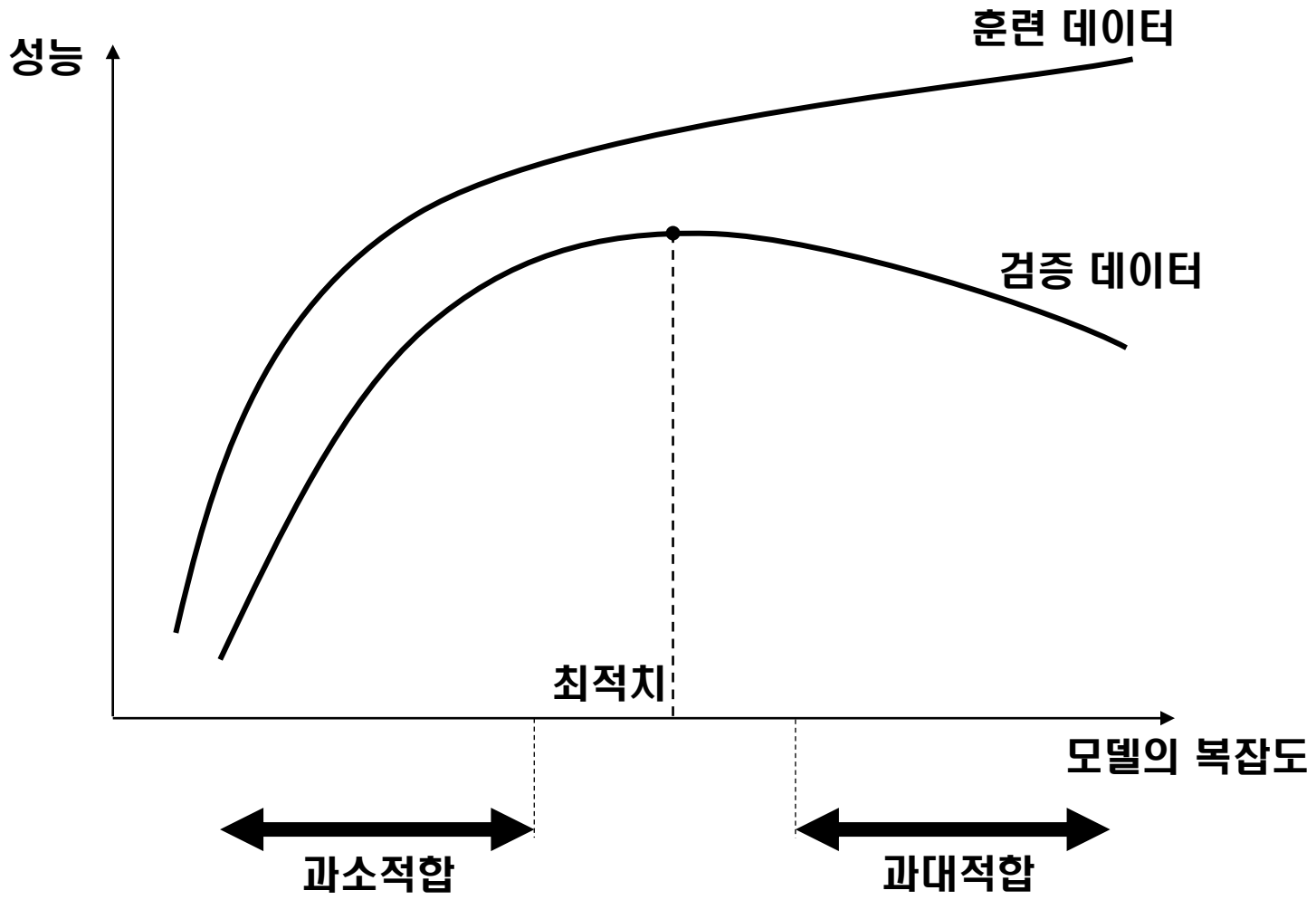
- k 값을 너무 작게 잡으면
  - 정확도가 올라갈 수 있으나
  - 주변 데이터(Noise)에 너무 예민하게 반응 → 과대 적합 가능성
- k 값을 너무 크게 잡으면
  - Noise에는 강하나
  - 주변에 너무 많은 데이터의 평균치를 사용하므로 정밀한 예측이 어려움
    → 과소 적합의 가능성
- 극단적으로 k=N(전체 샘플 수) 으로 하면
  - 항상 전체 데이터의 평균치 값을 예측
  - 영화 추천에서 이는 평균적으로 가장 많은 사람들이 본 영화 즉, 종합 베스트 셀러를 추천하는 것과 같은 결과

* Noise : 좋은 모델을 만드는데 도움이 되지 않는 혼란스러운 정보를 제공하는 샘플

과소 적합과 과대적합 판단 – 손실함수



과소 적합과 과대 적합 판단 – 성능



편향과 분산

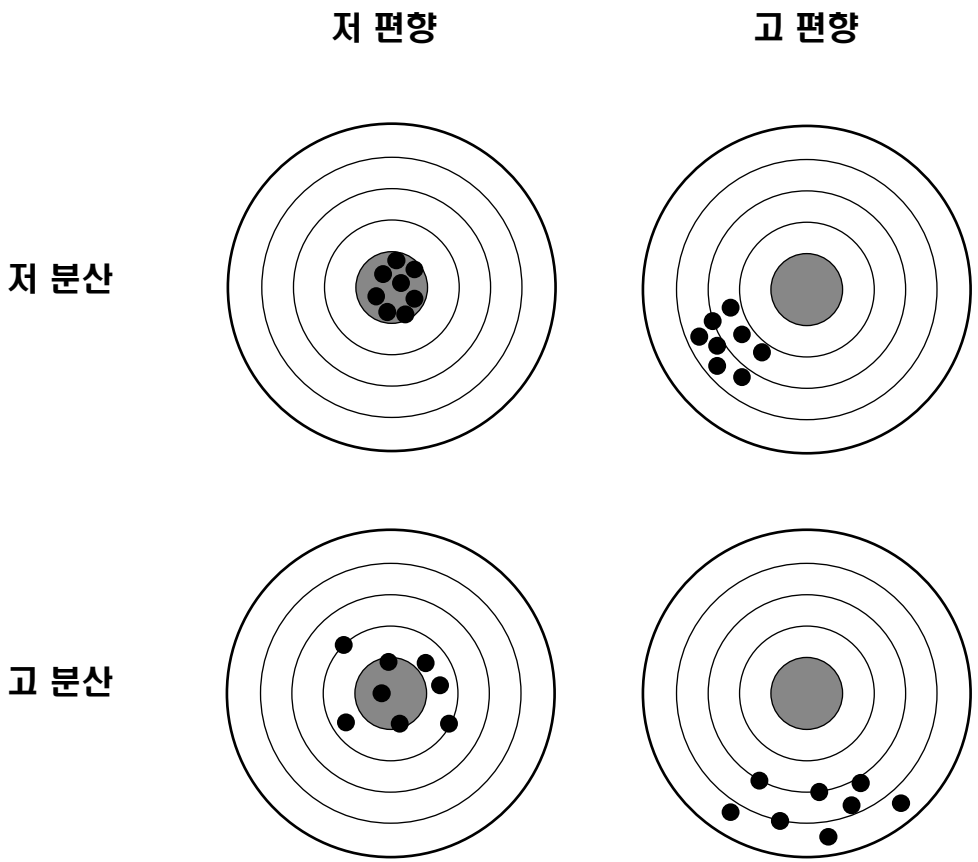
- 예측 모델에서 발생하는 오차는 분산(variance)과 편향(bias) 두가지 성분으로 설명할 수 있다.
- 분산이란 모델이 너무 복잡하거나 학습데이터 민감하게 반응하여 예측 값이 산발적으로 나타나는 것이다.
- 편향이란 모델 자체가 부정확하여 피할 수 없이 발생하는 오차를 말한다.

편향과 분산

- 예를 들어 kNN알고리즘에서 k=1인 경우는 모델이 학습 데이터에 민감하게 반응하여 예측의 변화가 커지므로 분산 성분이 증가한다.
  - 그러나 편향은 발생하지 않는다.
  - 이러한 경우를 과대적합되었다고 한다.

- 반면에 k=N인 경우는 모델은 항상 평균치로 동일하게 예측하므로 분산은 거의 발생하지 않는다.
  - 그러나 모델 자체가 부정확하여 편향이 커진다.
  - 모든 사람의 키를 평균치로 예측하면, 예측치는 평균치로 동일하므로 분산은 없어지지만 편향 오차가 클 수 밖에 없다.
  - 이는 과소적합의 현상이다.

편향과 분산



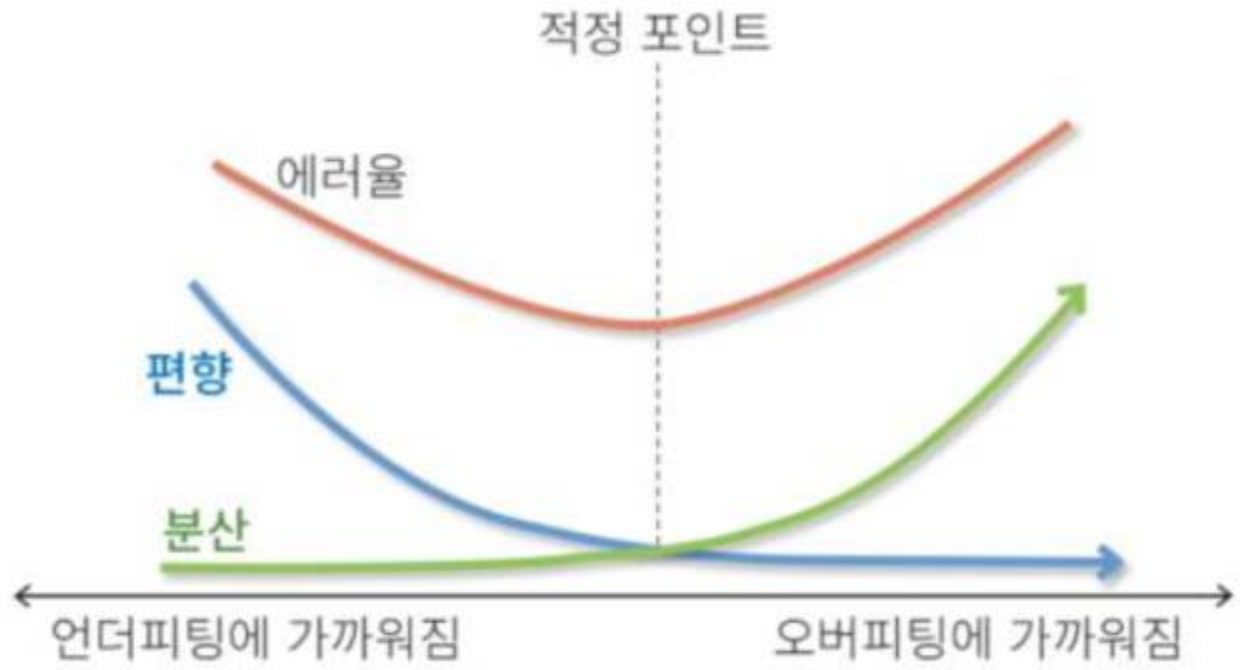
편향과 분산

- 모델이 훈련 데이터에 너무 종속적이거나 정교하면 (k=1 인 경우) 분산이 늘어나고 편향은 줄어든다 (위 그림에서 좌 하)

- 모델이 너무 단순하면 (k=N인 경우) 분산을 줄어드나 모델이 부족하여 생긴 편향이 증가한다.

- 뒤에서 설명할 결정 트리에서는 트리의 깊이(depth)에 따라서 편향과 분산 오류가 달라진다

머신 러닝



kNN 장단점

- kNN의 장점
  - 훈련시간이 거의 없다
  - 알고리즘의 개념이 명확
  - 모델링에 필요한 하이퍼 파라미터 : k 값 하나뿐
  - 특성 변수만 잘 선정하면 예측 성능도 좋다

- kNN의 단점
  - 분류를 처리하는 시간, 즉 알고리즘을 수행하는 시간이 길다
  - 확보한 샘플들을 모두 비교해서 어떤 그룹에 가까운 지 새로 계산해야
  - 또한 새로운 샘플이 계속 추가될 때마다 가까운 이웃이 달라진다
  - Lazy 알고리즘
    - 나중에 계산량이 많은 알고리즘

kNN 알고리즘의 변화

- 샘플들간의 거리에 대한 가중치를 고려
  - 가까이 있는 이웃에 대해서는 가중치를 크게

결정 트리

결정 트리 개요

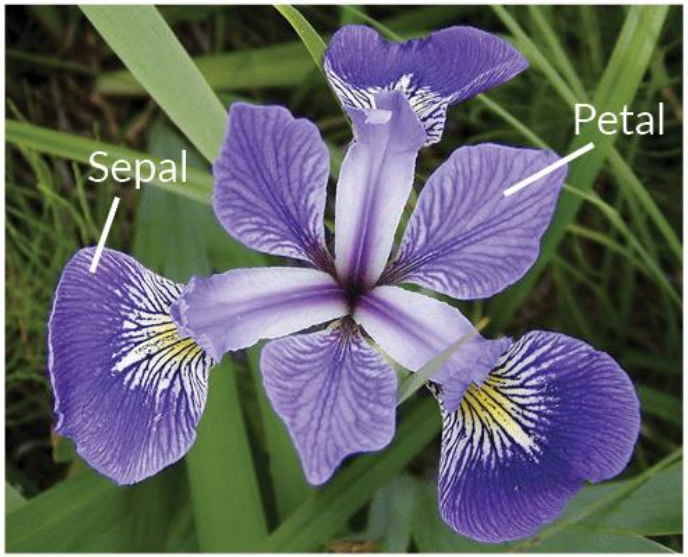
- **선형 모델**
  - **특성들을 대상으로 곱셈과 덧셈과 같은 연산**
  - **그 값을 기준으로 회귀나 분류를 예측**

- **결정 트리(**decision tree)
  - **각 특성을 독립적으로 하나씩 검토하여 분류 작업을 수행**
  - **마치 스무고개 하여 예측을 하듯이 동작 한 번에 한 특성을 따져보는 방법**
  - **분류, 회귀 모두에 사용**
    - **분류용 모델 :** DecisionTreeClassifier() **함수**
    - **회귀분석 모델 :** DecisionTreeRegressor() **함수**

동작 원리

- 결정 트리에서 핵심이 되는 부분
  - 가장 효과적인 분류를 위해서 먼저 어떤 변수를 가지고 판별을 할지 결정하는 것
- 이 판별은 트리를 내려가면서 계속
  - 매 단계마다 어떤 변수를 기준으로 분류를 하는 것이 가장 효과적인지를 찾아야 함
- 그룹을 효과적으로 “잘 나누는 것” 의 기준
  - 그룹을 나눈 후에 생성되는 하위 그룹들에 가능하면 같은 종류의 아이템들이 모이는지를 기준으로 삼는다
  - 한 그룹에 같은 종류의 아이템이 많이 모일수록 순수(pure)하다고 한다
  - 만일 나누어진 하위 그룹이 100% 같은 항목들로만 구성 → 순도(purity) 100%

결정 트리 예 – iris data

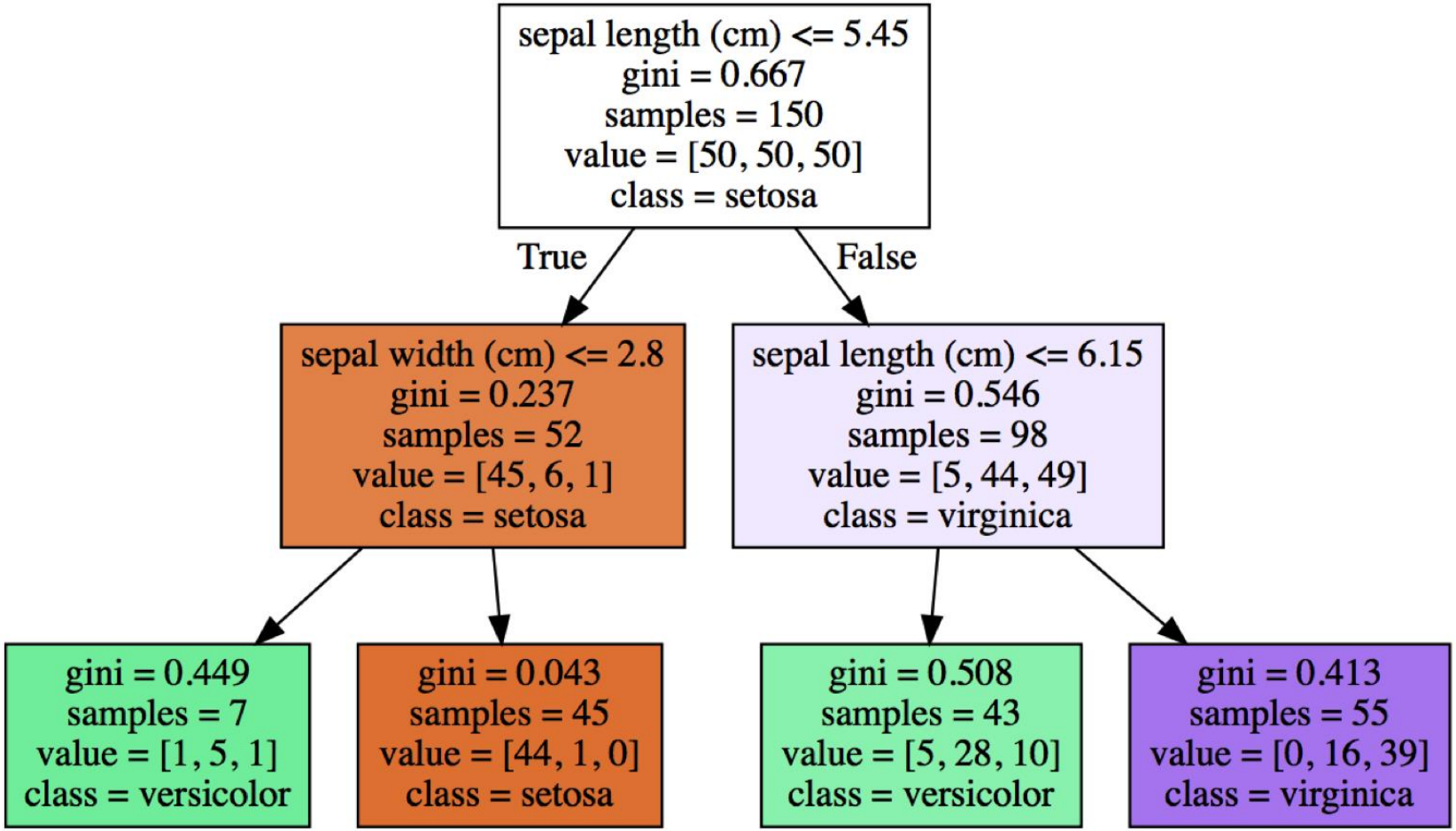


Iris Versicolor

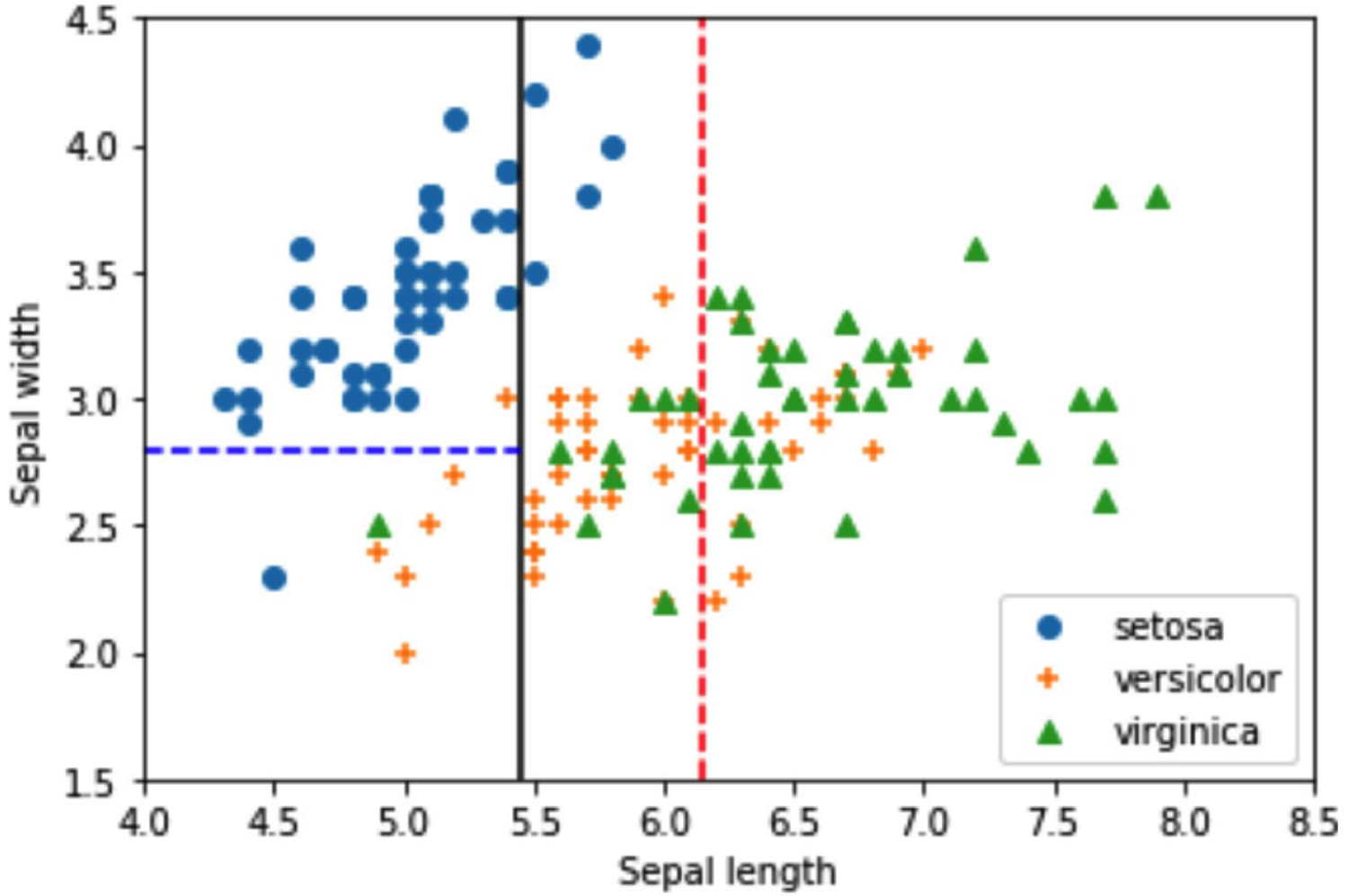
Iris Setosa

Iris Virginica

결정 트리 예


sepal length (cm) <= 5.45
gini = 0.667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
sepal width (cm) <= 2.8
gini = 0.237
samples = 52
value = [45, 6, 1]
class = setosa
sepal length (cm) <= 6.15
gini = 0.546
samples = 98
value = [5, 44, 49]
class = virginica
gini = 0.449
samples = 7
value = [1, 5, 1]
class = versicolor
gini = 0.043
samples = 45
value = [44, 1, 0]
class = setosa
gini = 0.508
samples = 43
value = [5, 28, 10]
class = versicolor
gini = 0.413
samples = 55
value = [0, 16, 39]
class = virginica

결정 트리 예

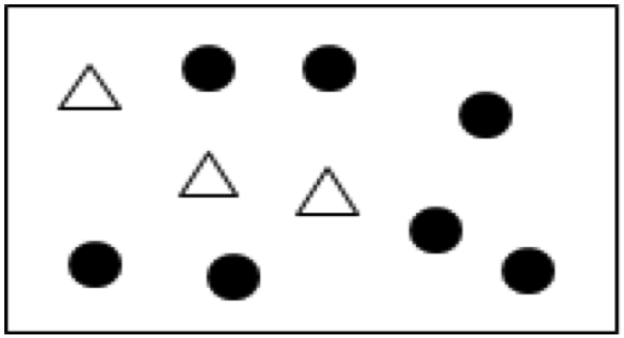

4.5
4.0
3.5
3.0
2.5
2.0
1.5
Sepal width
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
Sepal length
setosa
versicolor
virginica

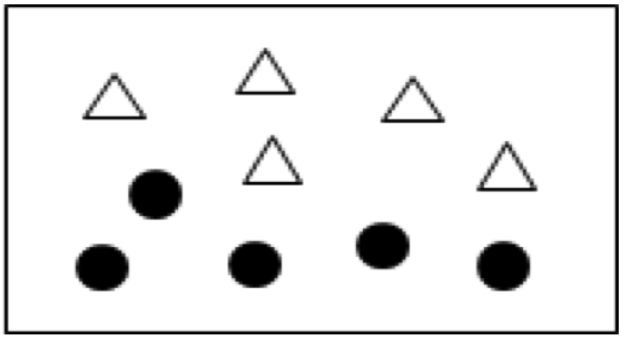
판별 기준

- **결정 트리는 나누어지는 그룹의 순도가 가장 높아지도록 그룹을 나누어야 한다**
- **그룹의 순도를 표현**
  - **지니(**Gini**) 계수, 엔트로피(**entrophy**) 주로 사용**
- Gini **계수**

$$Gini = 1 - \sum_{k=1}^{m} p_k^2$$

판별 기준





$$\text{좌측 박스: 지니}(7:3) = 1 - \left[\left(\frac{7}{10}\right)^2 + \left(\frac{3}{10}\right)^2\right] = 1 - (0.49 + 0.09) = 0.42$$

$$\text{우측 박스: 지니}(5:5) = 1 - \left[\left(\frac{5}{10}\right)^2 + \left(\frac{5}{10}\right)^2\right] = 1 - (0.25 + 0.25) = 0.5$$

정보량

- 데이터(이벤트)가 포함하고 있는 정보의 총 기대치, 정보의 가치
- 정보량을 표현
  - 해당 사건이 발생할 확률(probability)을 사용
- 사건 발생 확률에 따른 정보 가치
  - 확률 = 1 : 정보가 주는 가치가 없다
  - 사건 발생 확률이 낮을수록 : 정보가 주는 가치가 높음
- 정보량
  - 일어날 확률의 역수에 비례
- 정보량 정의

$$log\left(\frac{1}{p}\right)$$

정보량과 엔트로피

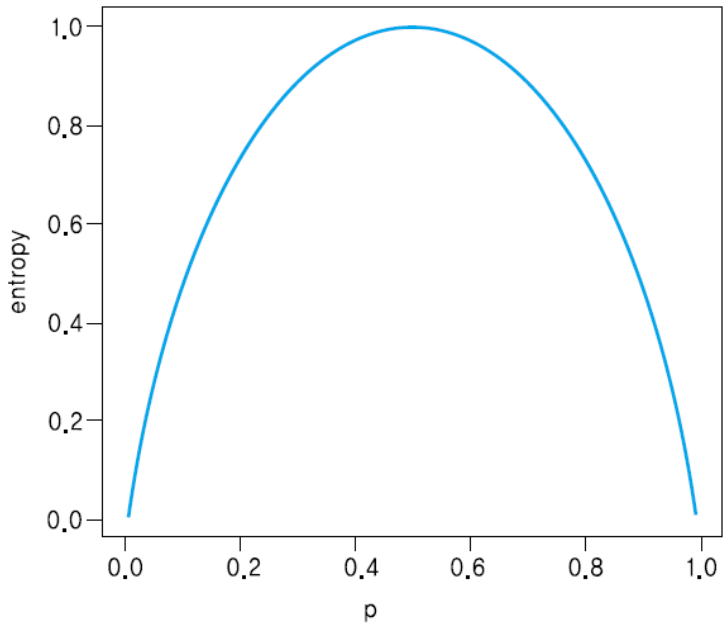
- 정보량의 기대치
  - 어떤 사건이 갖는 가치와 그 사건이 발생할 확률의 곱
  - 이를 엔트로피(entropy)라고 함

- 엔트로피(정보량의 기대치)

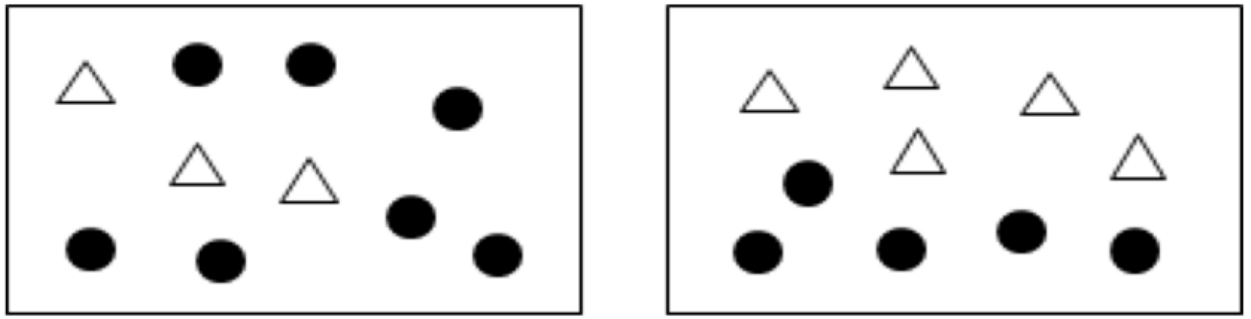
$$p \log\left(\frac{1}{p}\right) = -plog(p)$$

정보량과 엔트로피

- 바이너리(binary) 사건의 경우
  - 엔트로피는 p가 0.5 일 때 가장 높음
- 즉, 불확실성이 가장 높을 때 엔트로피가 가장 높음



정보량과 엔트로피



$$Entropy = -\sum_{k=1}^{m} p_k \log_2(p_k)$$

$$\begin{aligned} \text{좌측}(7:3) &= -\left[0.7^{*}\log_2(0.7) + 0.3^{*}\log_2(0.3)\right] \\ &= -\left[(-0.36) + (-0.52)\right] = -(-0.88) = 0.88 \end{aligned}$$

$$\begin{aligned} \text{우측}(5:5) &= -\left[0.5^{*}\log_2(0.5) + 0.5^{*}\log_2(0.5)\right] \\ &= -\left[(-0.5) + (-0.5)\right] = -(-1) = 1 \end{aligned}$$

지니 계수와 엔트로피

- 결정 트리의 성능
  - Gini 계수를 사용하든, entropy를 사용하든 성능에는 큰 차이가 없다

- 분류 속도(계산량)
  - Gini 계수의 계산 속도가 조금 빠르다 (entropy의 경우 log 연산 포함)

트리 종료 조건

- 결정 트리를 계속 만들어 상세하게 분류를 하면
  - 언젠가는 훈련 데이터에 대해서 100% 순도의 분류가 가능
  - 이는 과대적합 → 테스트 데이터에 대해서는 성능이 오히려 떨어지게 됨

- 결정 트리 모델의 트리 깊이
  - 깊이를 제한하지 않으면 과대적합할 위험이 높으므로 주의해야
  - 트리의 깊이를 적절한 값보다 너무 작게 제한하면 과소적합이 됨

트리 종료 조건 – 하이퍼 파라미터

- max_depth: **트리의 최대 깊이**
  - **이보다 깊은 트리를 만들지 않는다**
- max_leaf_nodes: **리프 노드의 최대 수**
  - **리프 노드를 이보다 많이 만들지 않는다**
- min_samples_split: **분할하기 위한 최소 샘플수**
  - **이보다 작으면 분할하지 않는다**
- min_samples_leaf: **리프 노드에 포함될 최소 샘플수**
  - **이보다 작은 노드는 만들지 않는다**
- max_features: **최대 특성수**
  - **분할할 때 이보다 적은 수의 특성만 사용한다**

내부 변수

- **결정 트리 모델을 만든 후에, 어떤 특성이 결정 트리를 생성할 때 중요한 역할을 했는지 비중을 파악 가능**
- **이 결과를 보고 중요하지 않은 특성은 향후에 제외하기도 함**
- **내부 변수로 확인**
  - feature_importances_

클래스 확률

- **테스트 결과**
  - **테스트 샘플이 속할 가장 확률이 높은 클래스 하나만 알려준다**
  - **이 샘플이 각 클래스에 속할 확률을 각각 알려주는 것도 가능**
- **소프트 투표(**soft voting**) 도입**
  - **보다 정확한 다중 분류를 수행할 수 있다**
  - **각 클래스에 속할 확률 :** predict_proba() **함수를 사용**

결정 트리의 특징

- 거의 모든 종류의 분류에 가장 많이 사용하는 범용 모델
- 장점
  - 알고리즘의 동작을 설명하기 수월함
    - 대출이 왜 거부되었는지
    - 당신의 신용도가 왜 낮은지
    - 왜 불합격되었는지 등
  - 특성 변수간의 연산이 없기 때문에 변수의 스케일링이 필요 없다
    - (분류 작업을 수행) 한번에 한 특성 변수씩 검토하여 어떤 기준으로 트리를 나누어 나가면 순도가 올라가는지만 점검하면 됨
- 단점
  - 훈련 데이터가 바뀌면 모델의 구조가 달라진다
  - 훈련 데이터에 따라 바뀌는 모델을 남에게 설명하기 어려울 수도

랜덤 포레스트

랜덤 포레스트(Random Forest)

- 결정 트리의 성능을 개선
- 비교적 간단한 구조의 결정 트리들을 수십~수백개를 랜덤하게 만들고 각 결정 트리의 동작 결과를 종합하여 판정
- 앙상블(ensemble) 방법
  - 여러 개의 모델을 만들고 평균을 구하는 방식
  - 하나의 모델만 만드는 것보다 좋은 성능을 보임
- 주어진 훈련 데이터를 모두 한 번에 사용해서 하나의 최상의 트리 모델을 만드는 방식이 아니라, 데이터의 일부 또는 속성의 일부만 랜덤하게 채택하여 결정 트리를 다양하게 만들고 그 결과의 평균치를 취하는 방식
- 나무(tree)가 많이 모였다는 의미로 숲(forest)라는 용어를 사용

개념 example

- 예1) 어떤 사람이 얼마나 훌륭한지를 평가
  - (방법 1)
    - 그 사람을 아는 모든 사람(예, 1,000명)에게 묻고 수집된 데이터로 한번에 평가
  - (앙상블 방법)
    - 그 사람을 아는 사람들을 랜덤한 조합(예, 50명씩 선택)으로 총 300개의 조합을 만들어 평가 → 이들 평가 점수의 평균치로 평가

- 예2) 나의 건강 정보에 대한 진단
  - (방법 1)
    - 나의 모든 건강 정보를 가장 훌륭한 의사 한 명에게만 주고 진단
  - (앙상블 방법)
    - 나의 모든 건강 정보의 일부를 랜덤하게 선택하는 것을 100번 수행하여 100명의 의사에게 각각 진단을 구하여 이들의 평균치(앙상블)로 최종 진단

- 일반적으로 앙상블 방법이 더 우수한 것으로 알려져 있다

특징

- 다수의 결정 트리를 생성하고 이의 평균을 구하면 단일 결정 트리를 사용하는 것보다 안정적이고 우수한 성능을 낸다.
  - 샘플도 랜덤하게 선택, 속성도 랜덤하게 선택
- 성능이 우수한 하나의 모델을 사용하는 것보다, 각각의 성능이 최상이 아니지만 다수의 모델을 사용하고 평균치를 구하는 방식이 더 우수
  - 이를 대중의 지혜, 큰 수의 법칙 등으로 설명하기도 함
- 단점
  - 모델의 동작을 한가지 트리를 선택하여 설명하기가 어렵다
  - 계산량이 많아진다

앙상블 기법

- 앙상블 기법
  - 여러 개의 작은 모델의 평균 값을 구하거나, 투표를 통하여 최적의 값을 찾는 절차 필요
  - 투표 방식
    - 직접 투표
    - 간접 투표

직접 투표와 간접 투표

- 직접 투표(hard voting)
  - 각 세부 모델이 예측한 최종 class에 1점 부여
  - 최종 target 변수
    - 가장 많은 점수를 받은 class

- 간접 투표(soft voting)
  - 각 세부 모델이 각 class에 속할 확률을 제공
  - 최종 target 변수
    - 각 세부 모델이 제공한 확률을 모두 더해서 가장 큰 값을 받은 class
  - (회귀 적용) 최종 예측 값
    - 각 세부 모델이 예측한 여러 수치 값의 평균치

- 어떤 투표 방식이 좋을까?
  - 문제의 성격에 따라 다르다

간접 투표(soft voting)

	P일 확률	Q일 확률	판정결과 (직접투표)
세부 모델 A	0.9	0.1	P
세부 모델 B	0.4	0.6	Q
세부 모델 C	0.3	0.7	Q
확률의 평균 (간접 투표)	(1.6)/3 = 0.533	(1.4)/3 = 0.456	P or Q

- **앙상블 방법의 회귀 분석 적용**
  - **최종 예측 값**
    - **각 세부 모델이 예측한 여러 수치 값의 평균치**

배깅

- **배깅(**bootstrap aggregation)
  - **전체 훈련 데이터에서 "중복을 허용" 하여 데이터를 샘플링을 하는 방법**
  - bootstrap resampling**의 줄임말로 부트 스트래핑이라고도 함**
  - **목적 : 부족한 훈련 데이터를 효과적으로 늘리기 위함**
- **페이스팅(**pasting)
  - **배깅과 달리 주어진 원래 데이터에서 중복을 허용하지 않고, 즉, 한 번 샘플링 된 것은 다음 샘플링에서 제외하는 방식**
- **배깅을 수행하면 학습에 선택되지 않는 샘플은 평균 37% 정도**
  - **이 샘플을** oob(out of bag) **샘플이라고 함**
  - **이** oob **데이터는 훈련에 사용되지 않았으므로 검증에 사용하기에 좋다**
- **랜덤 포레스트 모델**
  - **결정 트리 구조에 배깅을 적용한 방식**

배깅

- (참고) n개의 샘플에서 랜덤하게 하나를 선택할 때 어떤 샘플이 선택되지 않을 확률은

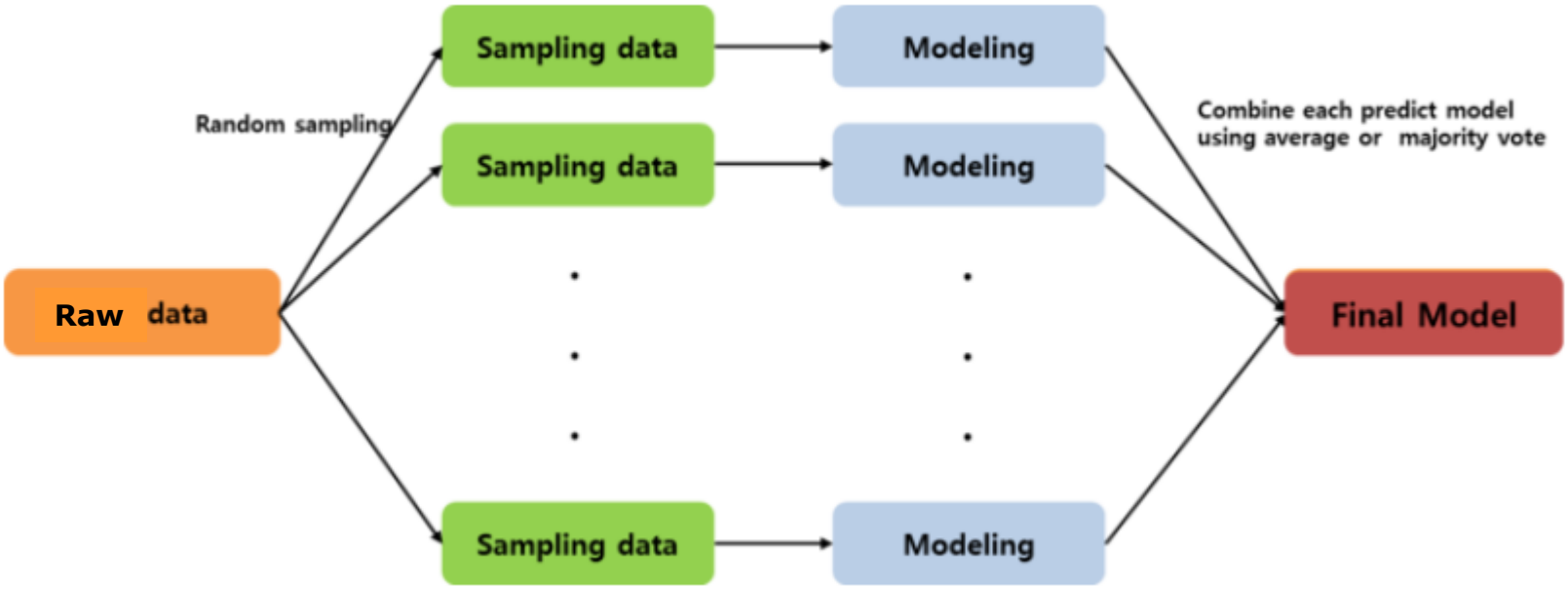
$$(1-\frac{1}{n})$$

이를 n번 시행해도 계속 선택되지 않을 확률은 독립 사건이 n 번 일어날 확률이므로

$$(1-\frac{1}{n})^n$$

n이 무한대로 커지면(즉, 샘플의 수가 커지면), 이 값은 1/e=0.3679 로 수렴한다.

배깅


Raw data
Random sampling
Sampling data
Sampling data
Sampling data
Modeling
Modeling
Modeling
Combine each predict model
using average or majority vote
Final Model

그리드 탐색

- 그리드 탐색
  - 여러가지 hyper parameter들의 조합에 대해서 가능한 경우를 모두 수행해 보고 가장 성능이 좋은 경우를 찾는 방식
  - 하이퍼 파라미터가 가질 수 있는 전체 범위를 몇 개의 구간으로 나누어 일일이 하나씩 점검
    - SVC 모델의 경우, gamma 변수와 C 변수의 값을 각각 5가지, 4가지로 나누고 총 4x5=20가지 경우를 시도
    - 이 중에서 최고의 score를 얻는 gamma와 C 값을 찾는다.
- 하이퍼 파라미터 예시
  - kNN: k값
  - 결정 트리: 트리의 깊이, 분류 조건, 분류를 위한 최소 샘플수
  - SVM: 커널 타입, 커널 계수, 규제화 파라미터(감마, C)
  - 랜덤포레스트: 트리수, 사용할 특성수, 분리 조건, 분류할 최소 샘플수
  - 그라디언트 부스팅: 트리수, 학습률, 트리 깊이, 분할할 조건, 분류할 최소 샘플 수

그리드 탐색

- 과대적합을 피하기 위한 규제화용 파라미터도 하이퍼 파라미터
  - 커널 SVM에서 감마 파라미터를 조절
  - 신경망에서는 드롭 아웃율 등
- GridSearchCV() 함수를 사용
  - 그리드 탐색을 하며 동시에 교차검증을 수행하여 예상되는 성능을 측정하기에 편리
  - GridSearchCV()로 생성한 모델 객체는 fit, predict, score 함수를 제공하며 fit을 호출할 때, 여러 파라미터 조합에 대해서 교차검증을 수행
  - best_estimator_ : 선택된 최적의 hyper parameter 값
  - best_score_ : 평가 점수

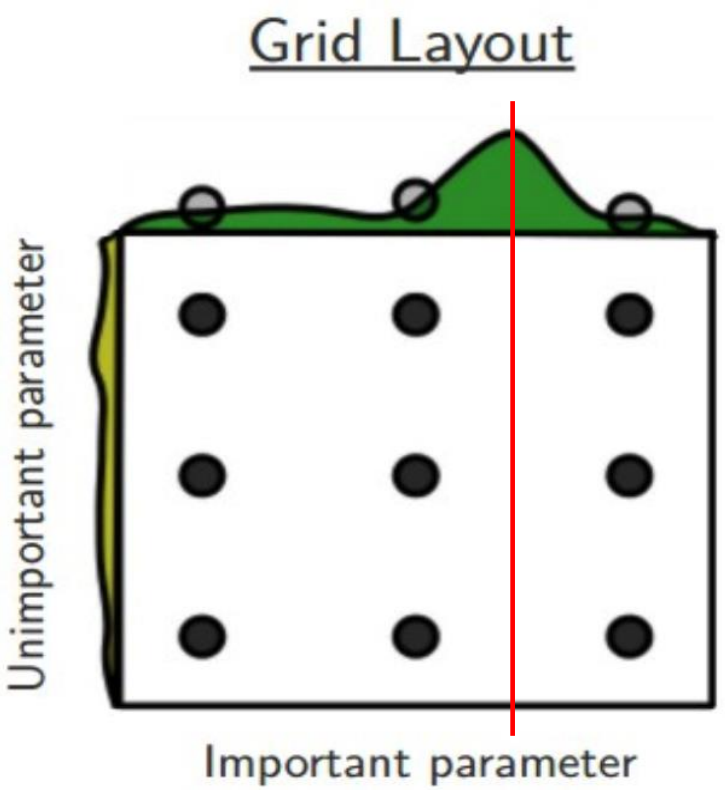
그리드 탐색

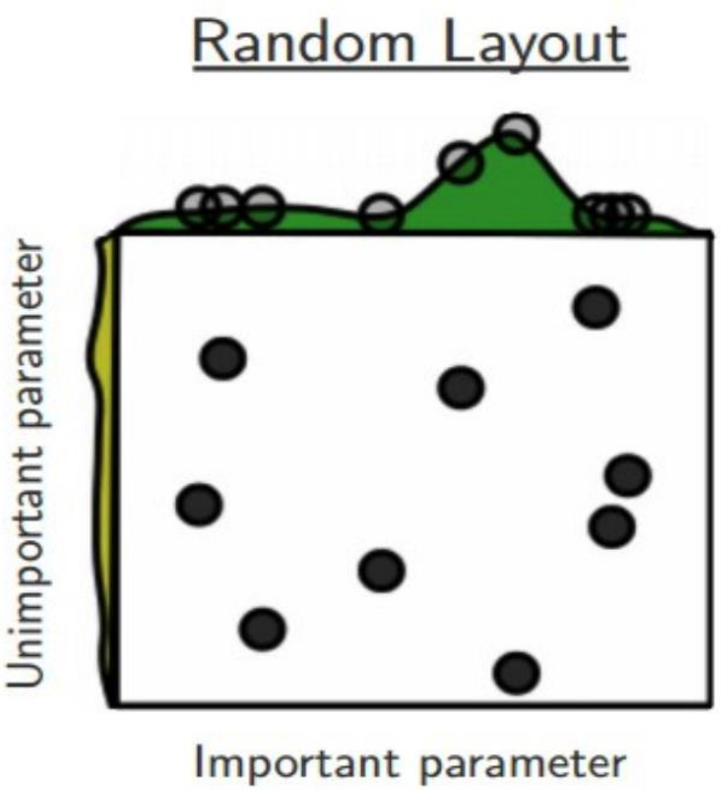
- 그리드 탐색의 특성
  - 단순하나
  - 여러 경우의 수를 모두 탐색하는데 시간도 오래 걸리고
  - 최적의 값을 놓치는 경우가 있다.
  - 격자 모양의 파라미터 조합의 값 사이에 실제로 최적의 하이퍼 파라미터 값이 있었다면 이를 찾아낼 방법이 없다.

랜덤 탐색

- 랜덤 탐색
  - 그리드 탐색의 단점 개선
    - 최적값을 놓치는 문제점
    - 탐색 시간도 감소
  - 두 단계로 수행
    - 1단계 : 일정한 범위 내에서 랜덤하게 하이퍼 파라미터를 선택하여 성능을 실험하여 대체로 어떤 영역에서 성능이 좋은지를 찾는다.
    - 2단계 : 다음에는 이 영역을 중심으로 세밀하게 탐색을 한다.
  - RandomizedSearchCV()함수를 사용

그리드 탐색 vs 랜덤 탐색



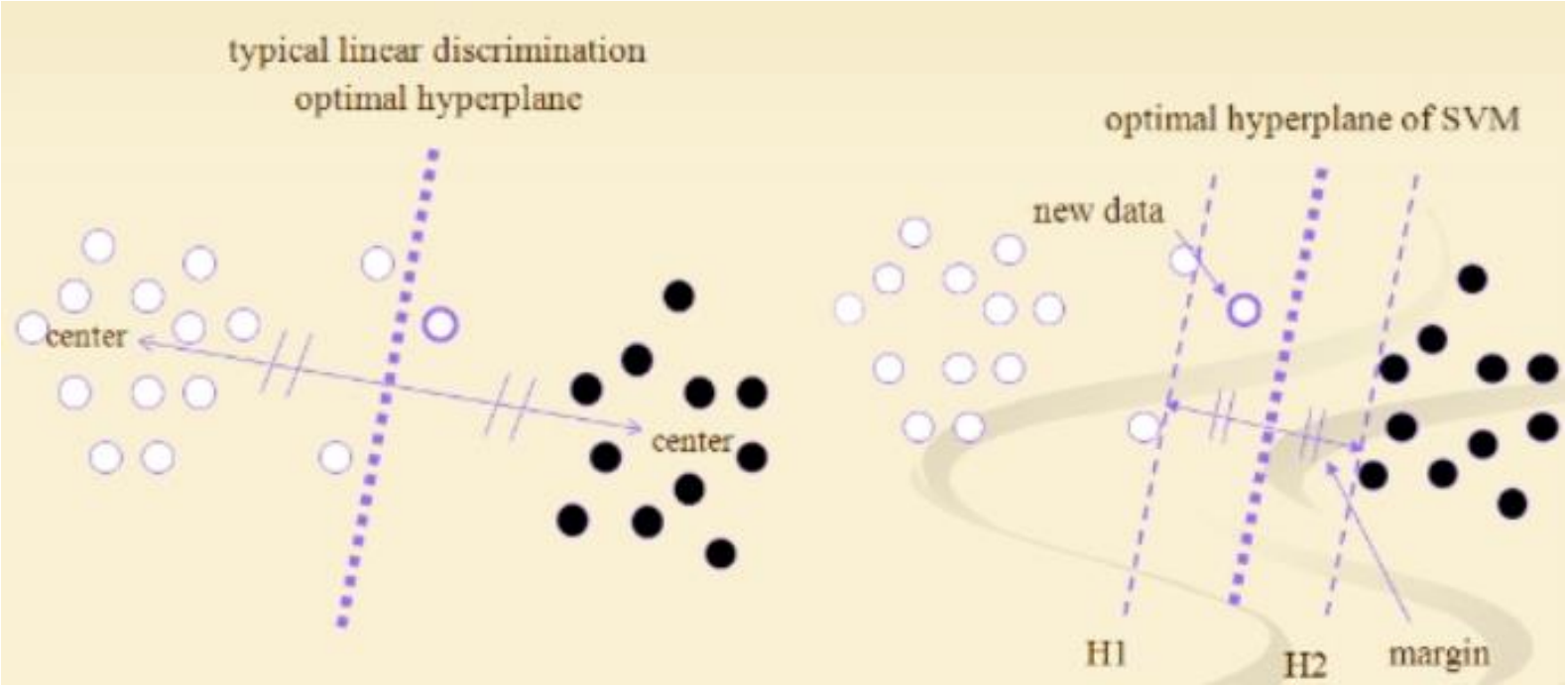


부스팅 알고리즘

- 부스팅 알고리즘
  - 앙상블 방법 중의 하나
  - 앞의 모델을 보고 성능을 순차적으로 점차 개선하는 방식으로 동작
  - 병렬로 처리하지 못함 (cf, 랜덤포레스트 : 병렬 처리 가능)
- 부스팅 알고리즘의 종류
  - 아다 부스트와 그라디언트 부스트가 널리 사용
- 아다 부스트(adaptive boosting)
  - 앞에서 사용한 세부 모델에서 과소 적합했던 샘플, 즉 분류에 실패한 샘플의 가중치를 높여주는 방법
  - 즉, 소외되었던 샘플을 주목하여 학습을 다시 시키는 방식

SVM

선형 분류 vs SVM


typical linear discrimination
optimal hyperplane
center
center
optimal hyperplane of SVM
new data
H1
H2
margin

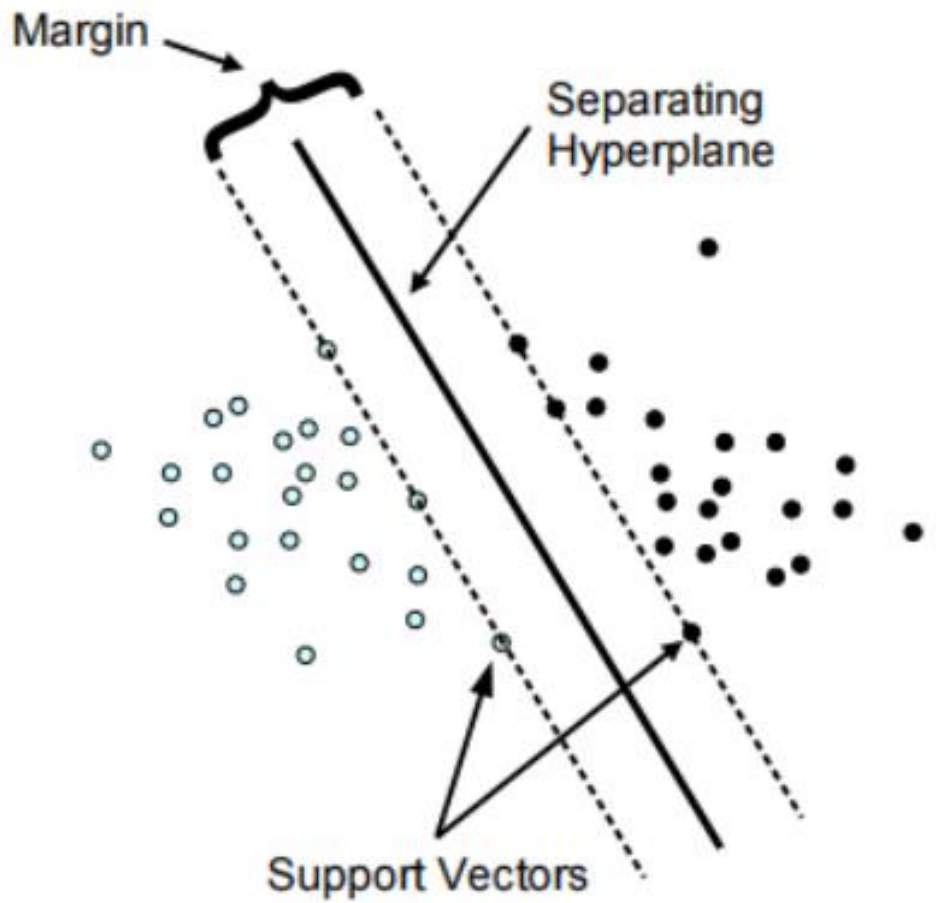
마진(Margin)

- support vector
  - 각 class의 data vector들로부터 판별 경계까지의 거리 중 가장 짧은 것

- margin
  - support vector와 판별 경계 사이의 거리

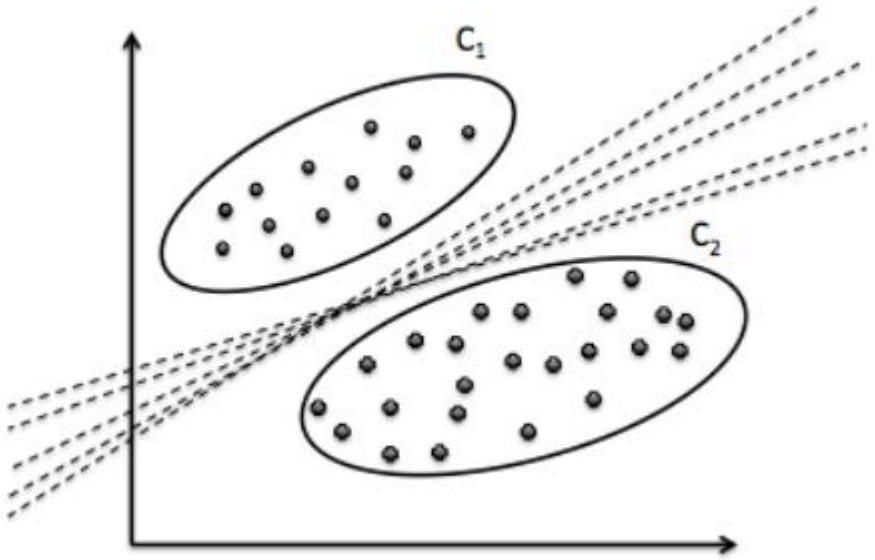
- margin이 클수록 → 분류를 잘 하는 것
  - 학습 데이터 중에서 noise를 무시하는 효과
    - 즉, 과대 적합을 피할 수 있다
  - 학습 데이터에 대해서 일정 오차를 내야 일반화 능력이 좋아진다

마진(Margin)



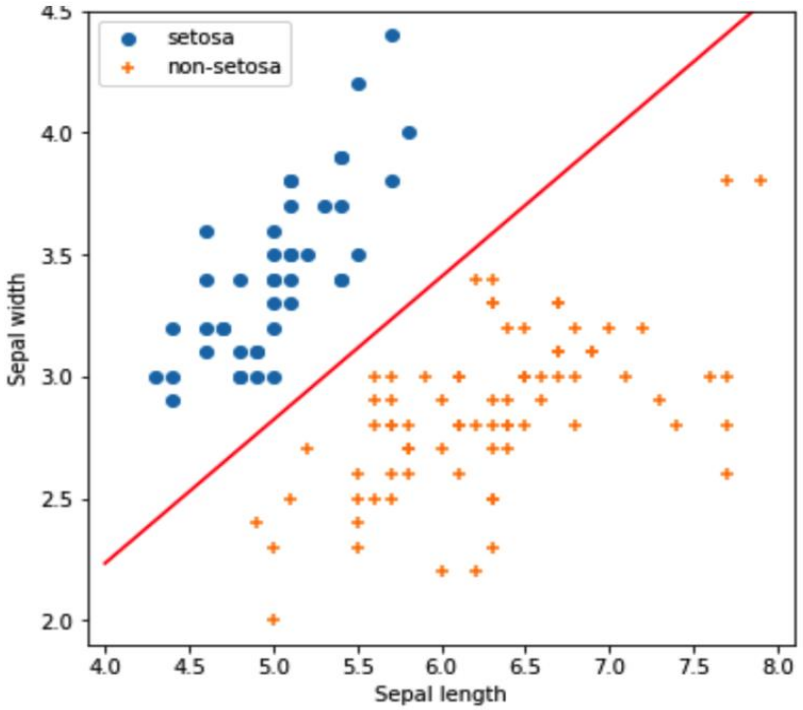
마진(Margin) – 판별 경계 선택 기준

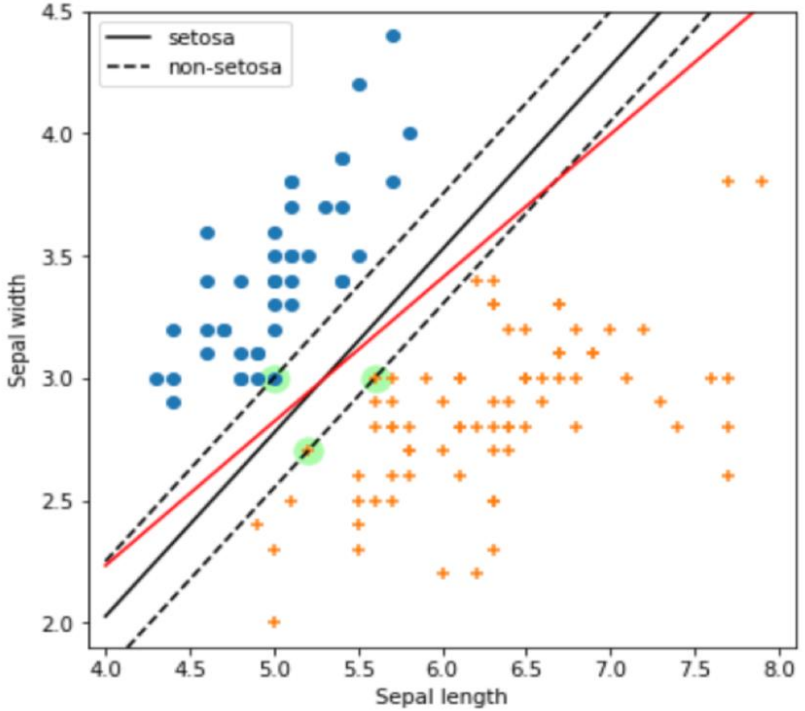
- 여러 개의 판별 경계 후보가 있을 때 선택 기준
  - margin : 넓은 것



서포트 벡터 머신(SVM)

- 선형 모델의 성능을 개선한 방법으로 널리 사용
- 분류 시에 결정 경계를 가능한 **마진을 넓게** 갖도록 만든 방식
  - 샘플들을 단순히 나누기만 하는 것이 아니라 가능한 거리를 멀리 나눌 수 있는 경계면을 찾는 작업을 한다





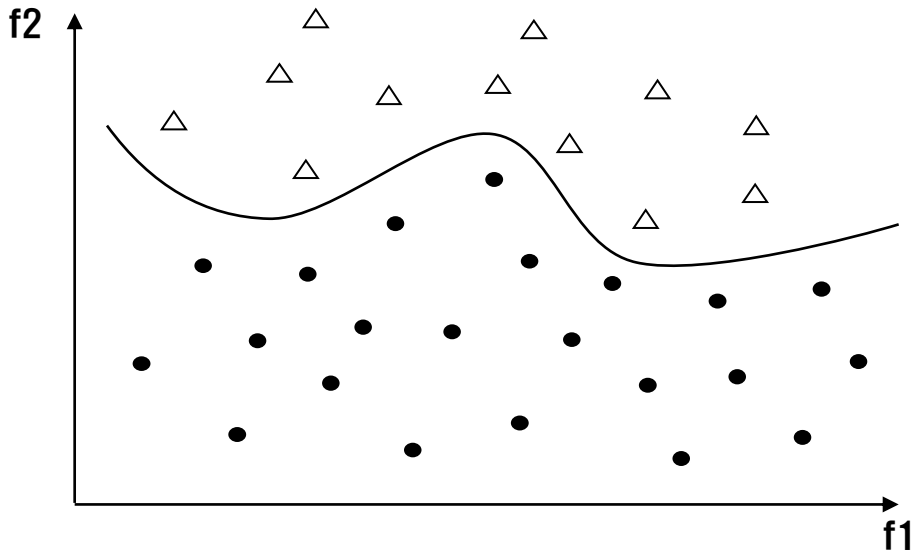
SVM의 특징

- 선형 분류 직선이 분류를 수행하는데 문제가 없어 보인다. 그러나 이러한 경계면은 새로운 샘플 데이터에 대해서는 잘 동작하지 않을 것이다
- 두께가 없는 선으로 경계를 만드는 것이 아니라 우측 그림의 점선으로 이루어진 두께가 있는 굵은 경계면을 만들고 그 두꺼운 경계면의 중앙을 지나는 선을 선택
  - 더 일반적인, 안정적인 경계선을 얻을 수 있으며 과대적합을 피할 수 있다

- (주의) SVM은 선형 모델과 마찬가지로 속성에 계수를 곱하고 덧셈을 하는 연산에 기반하므로 여러 속성을 함께 사용하려면 **반드시 스케일링**을 해야 한다.

커널 방식

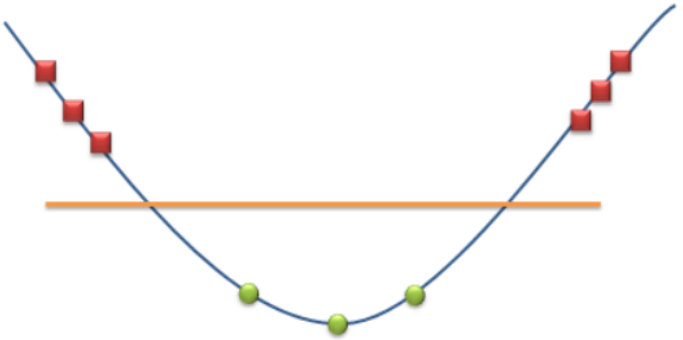
- SVM을 이용한 비선형 분류(kernel trick)
  - 선형 분리가 불가능한 저 차원의 특성을 그대로 사용하지 않고 이의 2승, 3승, 4승 등 고차원의 속성을 내부적으로 만들어서 사용하는 방식
  - RBF kernel이 가장 효율적인 것으로 알려져 있다
    - sklearn 모델의 기본 커널



커널 방식 – 개념

- **주어진 데이터 집합**
  - 저 차원의 특성을 가진 입력 샘플 공간

- **변형 데이터 집합**
  - 커널 트릭 사용 : 입력 샘플 공간을 고차원 특성 공간으로 변형
  - 직선으로 분류



- **주요 커널**
  - Linear Kernel, Polynomial Kernel, RBF(Radial Basis Function) Kernel

수고하셨습니다.

Q & A