인공지능과 신약개발을 위한 파이썬

6주차 머신 러닝의 이해

홍 성 은

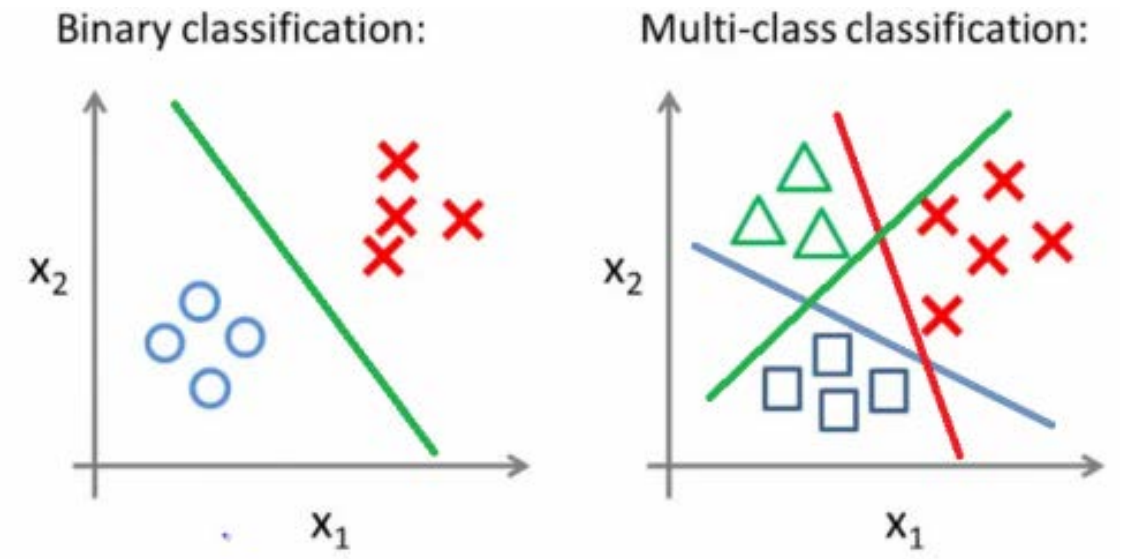
sungkenh@gmail.com

목차

- 분류
- 하이퍼파라미터 최적화
- 차원축소
- 클러스터링

분류

- 분류 문제
  - 새로운 데이터가 어떤 카테고리 집합에 속하는지 판단하는 것
  - 주어진 입력이 어떤 클래스(혹은 라벨)에 속하는지 예측하는 것

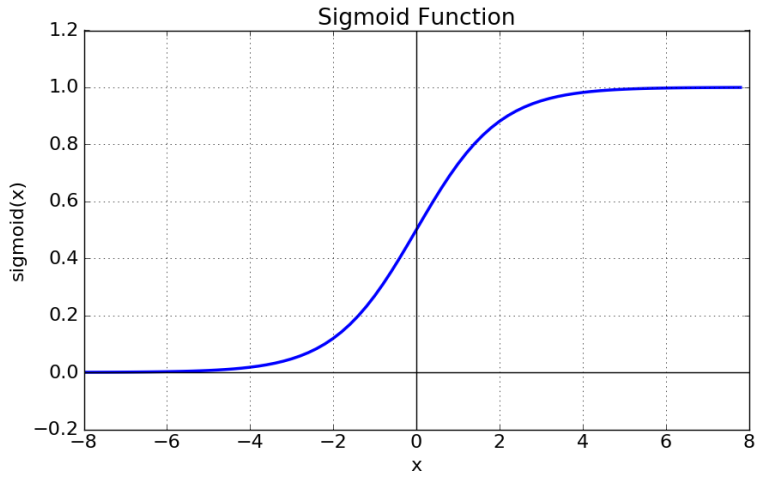


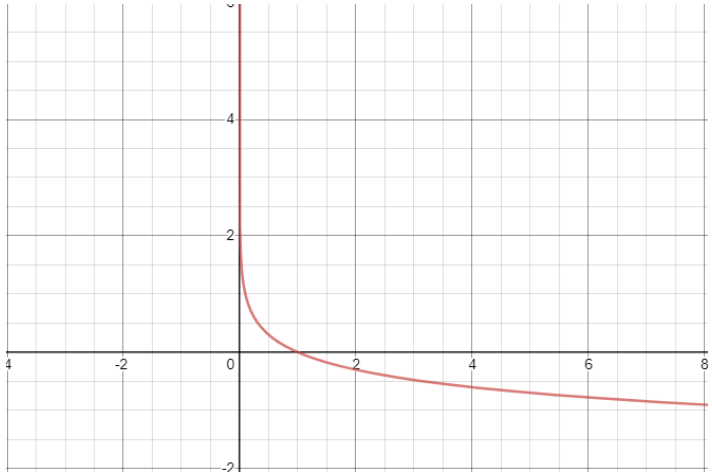
분류

- 분류에 사용되는 손실함수
  - BCE(Binary Cross Entropy): 이진 분류기를 훈련 시 사용하는 함수로 손실함수는 예측 값과 실제 값이 같으면 0이 되는 특성을 갖고 있어야 합니다. 예측 값과 실제 값이 모두 1로 같을 때 손실함수 값이 0이 되어야함

$$L = -\frac{1}{N}\sum_{i=1}^{N} t_i \log(y_i) + (1 - t_i)\log(1 - y_i)$$

$$\text{if } y_i = 1, t_i = 1, L = 0$$
$$\text{if } y_i = 0, t_i = 1, L = \infty$$



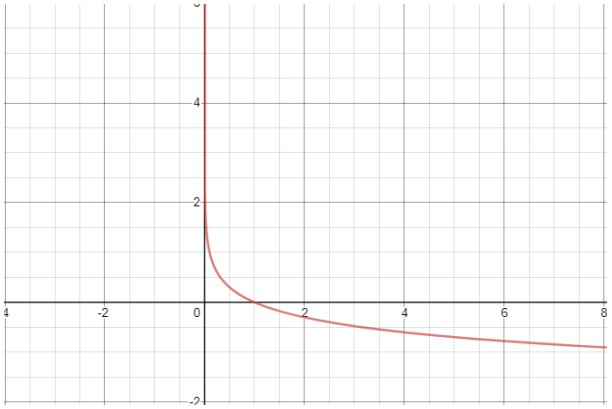


분류

- 분류에 사용되는 손실함수
  - Categorical Cross Entropy: 분류해야 할 클래스가 3개 이상인 경우, 즉 멀티클래스 분류에 사용(C는 클래스 갯수)
  - 라벨이 [0,0,1,0,0], [1,0,0,0,0], [0,0,0,1,0]과 같이 one-hot 형태로 제공될 때 사용

$$L = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{C} t_i \log(y_{ij})$$

  - 실제값과 예측값이 모두 [1 0 0 0 0] L= 0
  - 실제값은 [1 0 0 0 0], 예측값은 [0 1 0 0 0]인 경우 L=∞

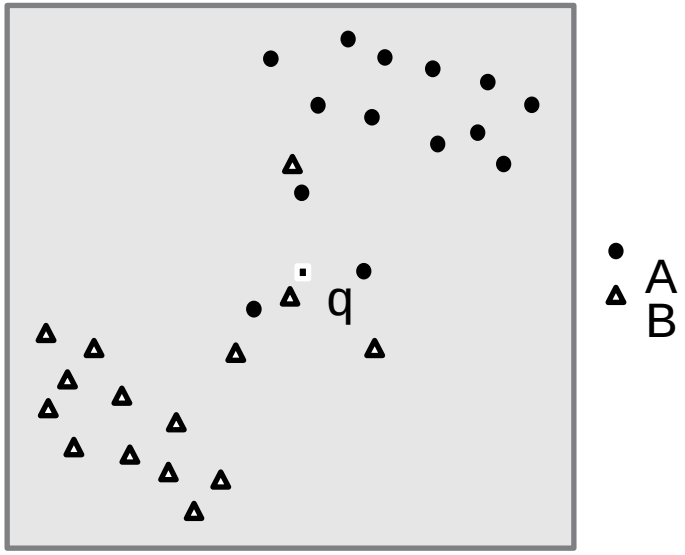


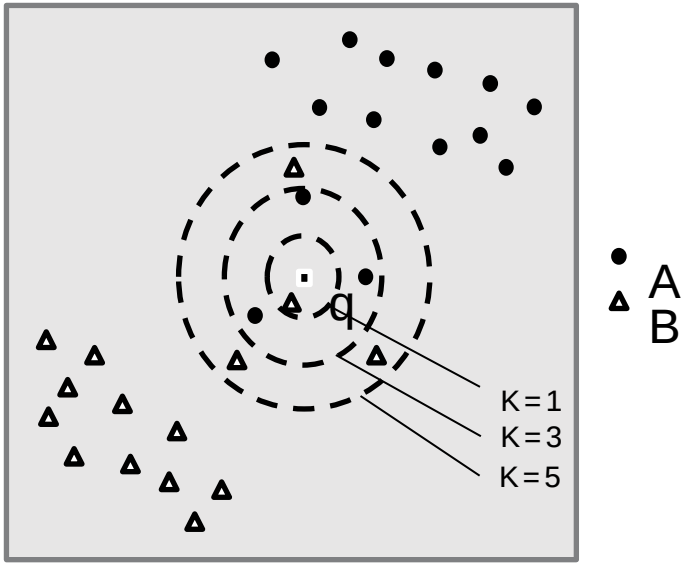
$$p_j = \frac{e^{\wedge_j}}{\sum_{k=1}^{K} e^{x_k}} = \frac{e^{x_j}}{e^{x_1}+e^{x_2}+\cdots+e^{x_K}} \quad for\ j\ =\ 1,\ldots,K$$

...(공식1: softmax 함수)

분류

- K-NN(K-nearest neighbor)
  - 주어진 샘플의 특성 값을 보고 가장 가까운 특성을 가지는 이웃(neighbor)을 k개 선택하고 이들 레이블의 평균치로 이 샘플이 속할 분류를 예측하는 방식
  - kNN은 직관적으로 이해하기 쉬운 분류 알고리즘으로서 추천 시스템에서 많이 사용됨
    - 적절한 추천을 하기 위해서 추천을 요청한 사람의 성향을 특성들로 파악하고 그 사람과 가장 성향이 유사한 k명의 사람들이 좋아하는 품목을 추천하는 방식을 사용
  - kNN알고리즘을 협업 필터링(collaborative filtering)이라고도 부름





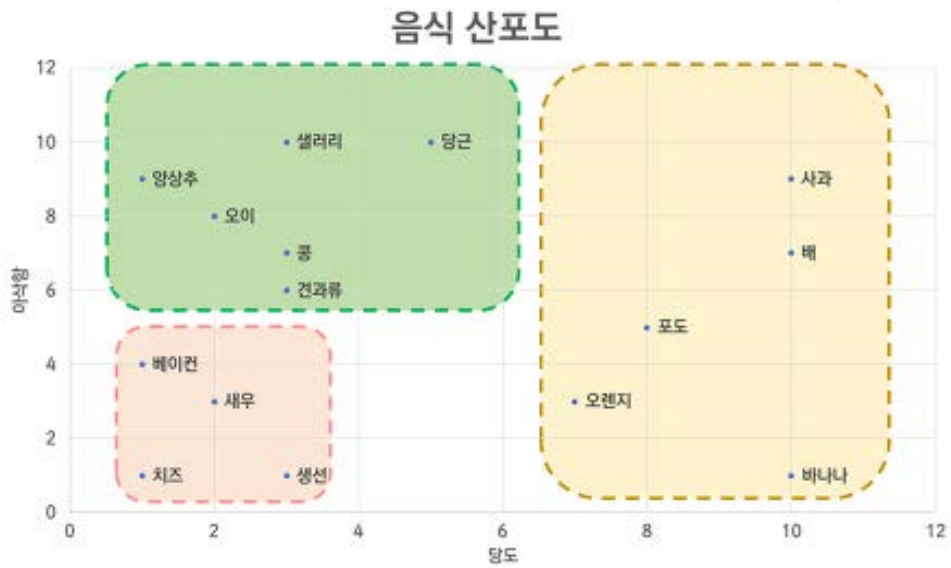
분류

- K값의 변화
  - k 값을 너무 작게 잡으면 주변 데이터에 너무 예민하게 반응하고 k 값을 너무 크게 잡으면 주변에 너무 많은 데이터의 평균치를 사용하므로 분류가 무뎌짐
  - 극단적으로 k=N (전체 샘플 수)로 잡으면 항상 전체 데이터의 평균치 값을 예측하게 됨
    - 영화 추천에서 k=N으로 한다면 이는 평균적으로 가장 많은 사람들이 본 영화 즉, 종합 베스트셀러를 추천하는 것과 같음
  - k값을 작게 잡으면 노이즈에 민감하나 정확도는 올라가고 k를 크게 잡을수록 노이즈에 강하나 정밀한 예측이 어려움
  - kNN의 단점은 훈련시간이 거의 없는 것에 비해 분류를 처리하는 시간, 즉 알고리즘을 수행하는 시간이 길다는 것임

분류

- K-NN예제

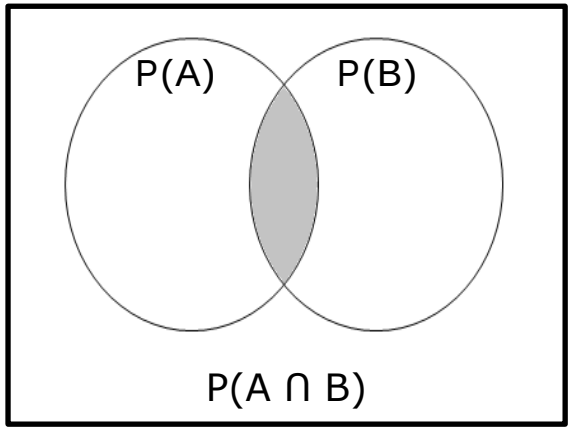
	이름	당도	아삭함	종류
1	사과	10	9	과일
2	베이컨	1	4	단백질
3	바나나	10	1	과일
4	당근	5	10	채소
5	샐러리	3	10	채소
6	치즈	1	1	단백질
7	오이	2	8	채소
8	생선	3	1	단백질
9	포도	8	5	과일
10	콩	3	7	채소
11	양상추	1	9	채소
12	견과류	3	6	단백질
13	오렌지	7	3	과일
14	배	10	7	과일
15	새우	2	3	단백질
16	토마토	6	4	???



이름	당도	아삭함	종류	거리	k	토마토결과
오렌지	7	3	과일	2	1	과일
포도	8	5	과일	5	2	과일
견과류	3	6	단백질	13	3	과일
새우	2	3	단백질	17	4	단백질
생선	3	1	단백질	18	5	단백질
콩	3	7	채소	18	6	단백질
베이컨	1	4	단백질	25	7	단백질
바나나	10	1	과일	25	8	단백질
배	10	7	과일	25	9	과일
오이	2	8	채소	32	10	단백질
치즈	1	1	단백질	34	11	단백질
당근	5	10	채소	37	12	단백질
사과	10	9	과일	41	13	단백질
샐러리	3	10	채소	45	14	과일
양상추	1	9	채소	50	15	채소

분류

- 베이즈 정리
  - 조건부 확률$P(A|B)$은 **사건B가 발생한 경우 사건 A의 확률**
  - 베이즈 정리는 $P(A|B)$ 의 추정이 $P(A \cap B)$와 $P(B)$에 기반을 두어야한다는 정리
  - $P(A|B) = \frac{P(A \cap B)}{P(B)}$



  - 전체 사건 중 비가 온 확률 P(비) = $\frac{7}{20}$, P(비가안옴) = $\frac{13}{20}$
  - P(비|맑은날) = $P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$
  - P(맑은날|비), P(비), P(맑은날) = $\frac{2}{7}$, $\frac{7}{20}$, $\frac{10}{20}$ = $\frac{\left(\frac{2}{7}\right) * 0.35}{0.5}$ = 0.2

날씨와 비의 관계표			
	비왔는가?		
	옴	안옴	
맑은날	2	8	10
흐린날	5	5	10
	7	13	20

날씨와 비의 관계표			
	비왔는가?		
	옴	안옴	
맑은날	P(맑은날|비) = 2/7 = 0.28…	8	P(맑은날) = 10/20 = 0.5
흐린날	5	5	10
	P(비) = 7/20 = 0.35	13	20

분류

- 나이브(Naive) 베이즈(Bayes)
  - 데이터셋의 모든 특징들이 동등하고 독립적이라고 가정
  - 비가 오는날에는 시간보다 습도가 더 중요한 변수가 될 수 있지만 이런 사실을 모두 무시함
  - 이런 가정에도 분류에서 매우 정확한 값을 내놓기에 자주 사용되고 있음

비가 올 확률

$$= \frac{P(\text{비} \mid \text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도})}{P(\text{비} \mid \text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도}) + P(\sim\text{비} \mid \text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도})}$$

구해야 할 것

- $P(\text{비} \mid \text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도})$
- $P(\sim\text{비} \mid \text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도})$

비 와 변수들간 관계표

	날씨가 좋은가?		바람이 많이 부는가?		기압이 높은가?		온도가 높은가?		
	Yes	No	Yes	No	Yes	No	Yes	No	계
비 온날	2	6	6	2	8	0	5	3	8
안온날	8	4	2	10	2	10	6	6	12
계	10	10	8	12	10	10	11	9	20

$$P(\text{비} \mid \text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도})$$

$$= \frac{P(\text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도} \mid \text{비})P(\text{비})}{P(\text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도})}$$

$$= \frac{P(\text{날씨}|\text{비})\,P(\sim\text{바람}|\text{비})\,P(\text{기압}|\text{비})\,P(\sim\text{온도}|\text{비})P(\text{비})}{P(\text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도})}$$

$$\fallingdotseq P(\text{날씨}|\text{비})\,P(\sim\text{바람}|\text{비})\,P(\text{기압}|\text{비})\,P(\sim\text{온도}|\text{비})P(\text{비})$$

1. $P(\text{날씨}|\text{비})\,P(\sim\text{바람}|\text{비})\,P(\text{기압}|\text{비})\,P(\sim\text{온도}|\text{비})P(\text{비})$

$= (2/8) * (2/8) * (8/8) * (3/8) * (8/20)$

$= 0.009375$

2. $P(\text{날씨}|\sim\text{비})\,P(\sim\text{바람}|\sim\text{비})\,P(\text{기압}|\sim\text{비})\,P(\sim\text{온도}|\sim\text{비})P(\sim\text{비})$

$= (8/12) * (10/12) * (2/12) * (6/12) * (12/20)$

$= 0.33333$

분류

- 나이브(Naive) 베이즈(Bayes)

비가 올 확률

$$= \frac{P(\text{비} \mid \text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도})}{P(\text{비} \mid \text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도}) + P(\sim\text{비} \mid \text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도})}$$

$$= \frac{0.009375}{0.009375+0.333333} = 0.027 \rightarrow 2.7\%$$

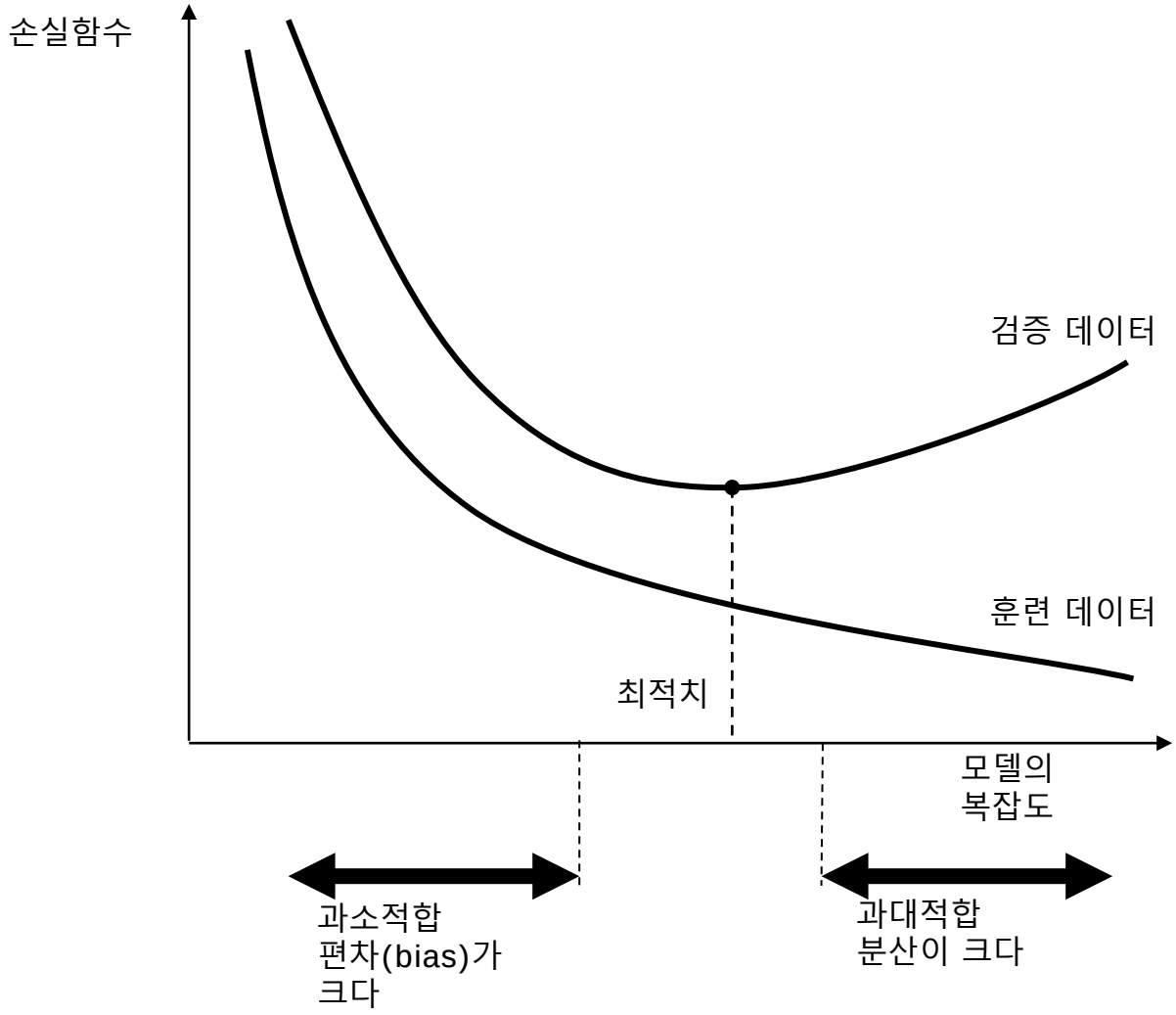
비가 안 올 확률

$$= \frac{P(\sim\text{비} \mid \text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도})}{P(\text{비} \mid \text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도}) + P(\sim\text{비} \mid \text{날씨} \cap \sim\text{바람} \cap \text{기압} \cap \sim\text{온도})}$$

$$= \frac{0.333333}{0.009375+0.333333} = 0.973 \rightarrow 97.3\%$$

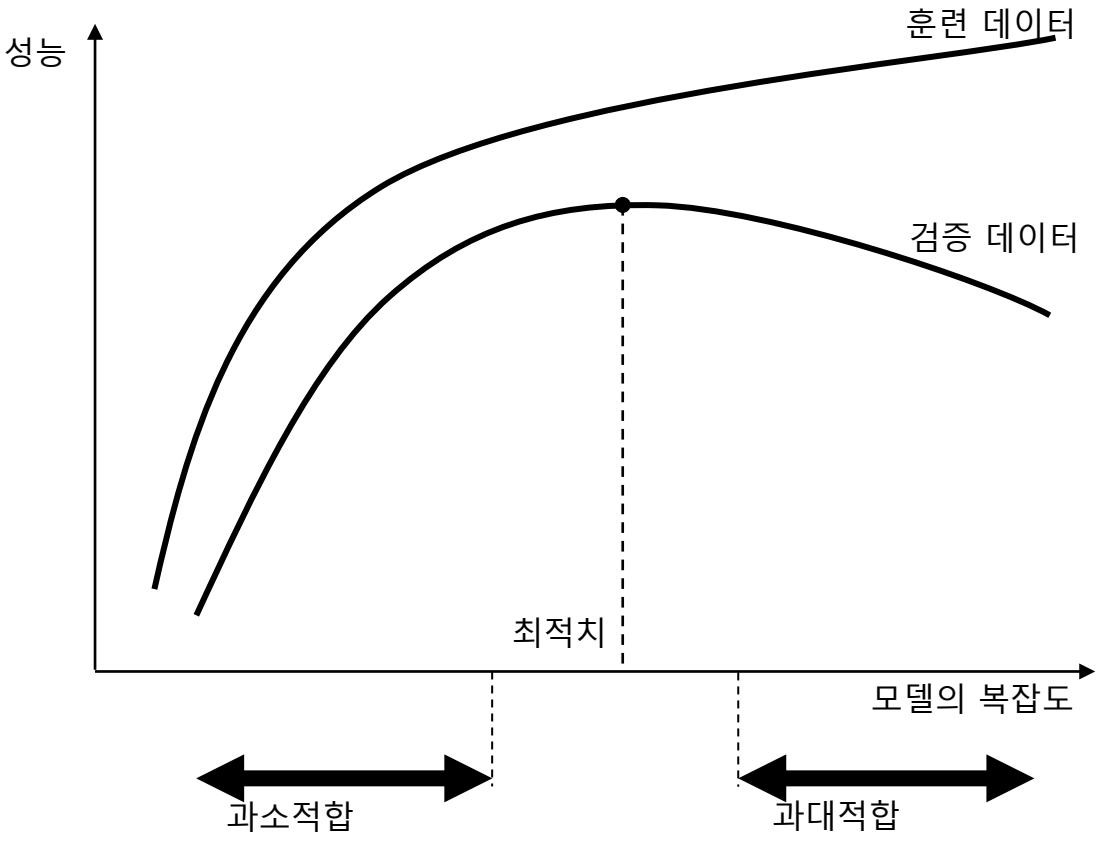
분류

- 손실함수 그래프로 과적합 판단



분류

- 성능 그래프로 과적합 판단



분류

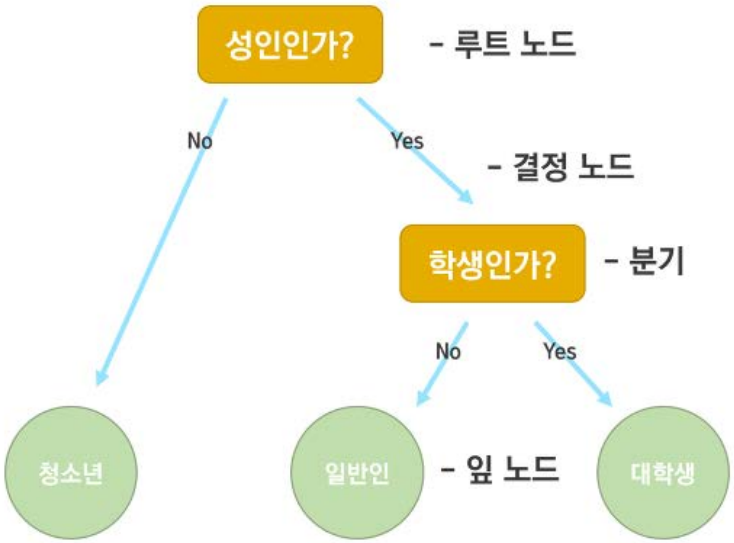
- 결정 트리
  - 분류 기법 적용에서 가장 많이 사용하는 방법
  - 분류 작업을 수행하기 위해 한번에 한 특성 변수를 해석
  - 결정 트리 모델을 사용하면 동작을 설명하기 수월함
    - 대출 거부 사유
    - 신용도가 낮은 이유
    - 불합격 사유 등
- 결정 트리 특징
  - 선형회귀 모델은 특성들을 대상으로 곱셈과 덧셈과 같은 연산을 하고 그 값을 기준으로 회귀나 분류를 예측했음
  - 결정 트리(decision tree)는 이와 달리 각 특성을 독립적으로 하나씩 검토하여 분류 작업을 수행함
    - 마치 스무고개 하여 예측을 하듯이 동작 한 번에 한 특성을 따져보는 방법임
  - 결정 트리는 주로 분류와 회귀에 모두 사용된다.
    - 분류용 모델은 DecisionTreeClassifier가 있고 회귀분석 모델로는 DecisionTreeRegressor가 제공됨

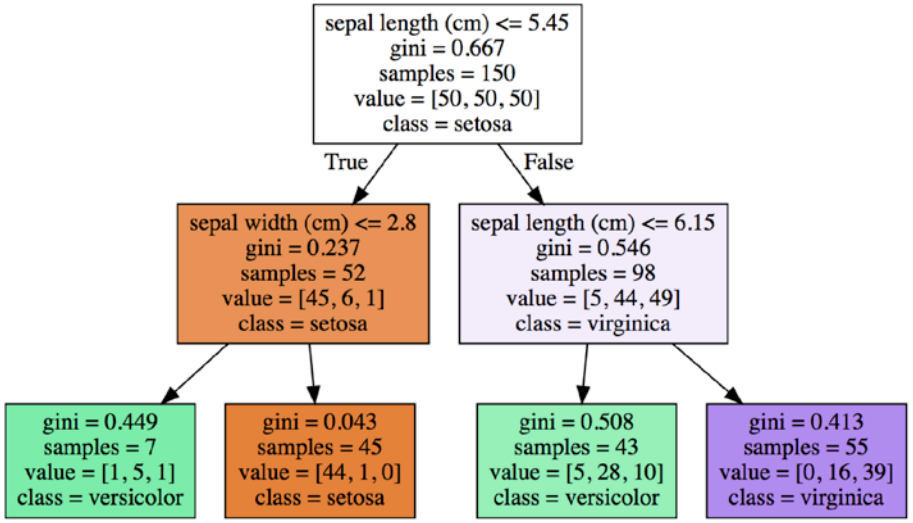
분류

- 동작 원리
  - 결정 트리에서 핵심이 되는 부분은 가장 효과적인 분류를 위해서 먼저 어떤 변수를 가지고 판별을 할지 결정하는 것
  - 이 판별은 트리를 내려가면서 계속되어야 하는데 매 단계마다 어떤 변수를 기준으로 분류를 하는 것이 가장 효과적인지를 찾아야 함
  - 여기서 그룹을 효과적으로 "잘 나누는 것"의 기준은 그룹을 나눈 후에 생성되는 하위 그룹들에 가능하면 같은 종류의 아이템들이 모이는지를 기준으로 삼음
  - 한 그룹에 같은 종류의 아이템이 많이 모일수록 순수(pure)하다고 하는데, 만일 나누어진 하위 그룹이 100% 같은 항목들로만 구성되면, 순도(purity)가 100%라고 함

분류

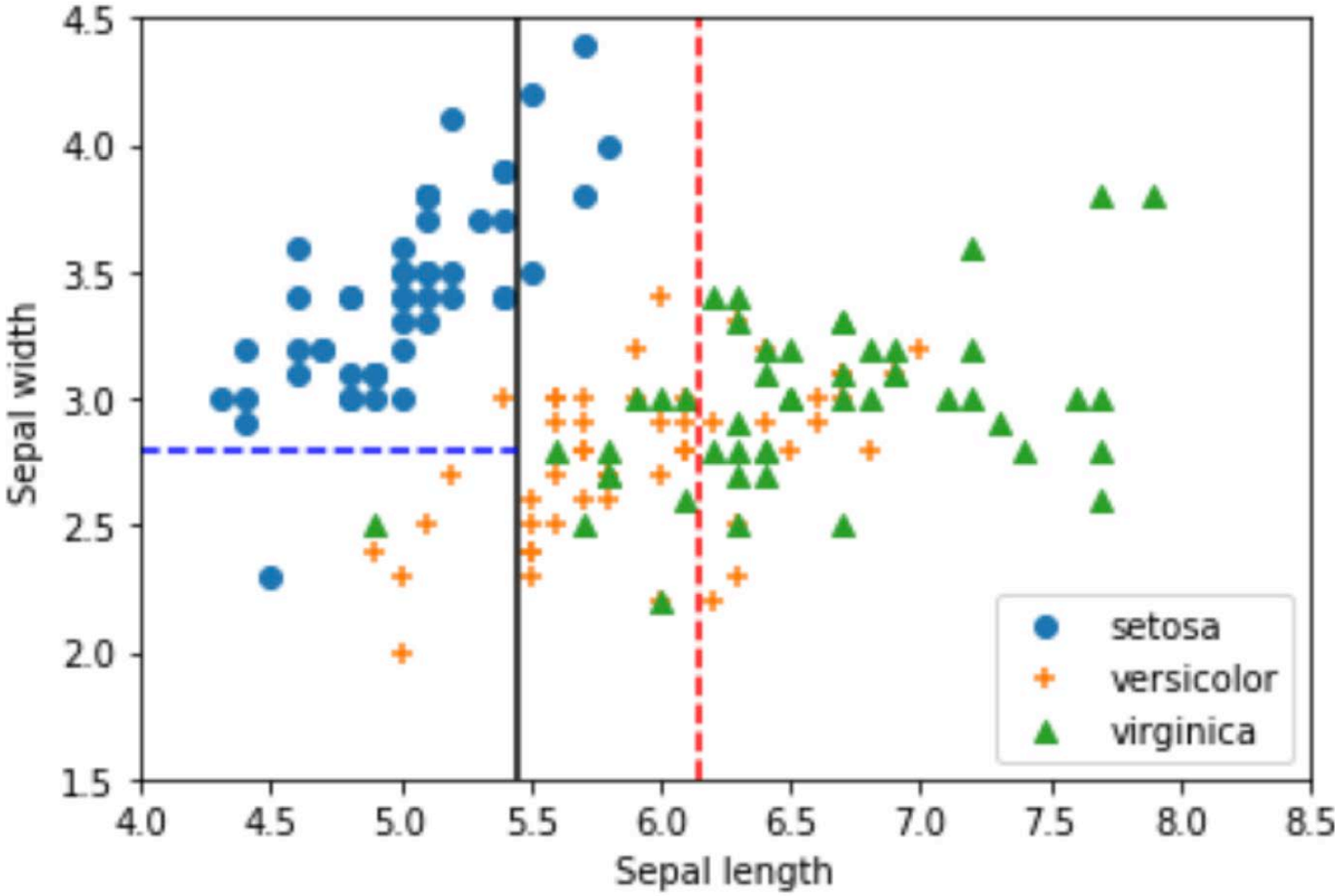
- 결정트리 예


성인인가?
- 루트 노드
No
Yes
- 결정 노드
학생인가?
- 분기
No
Yes
청소년
일반인
- 잎 노드
대학생


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

분류

- 결정트리 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

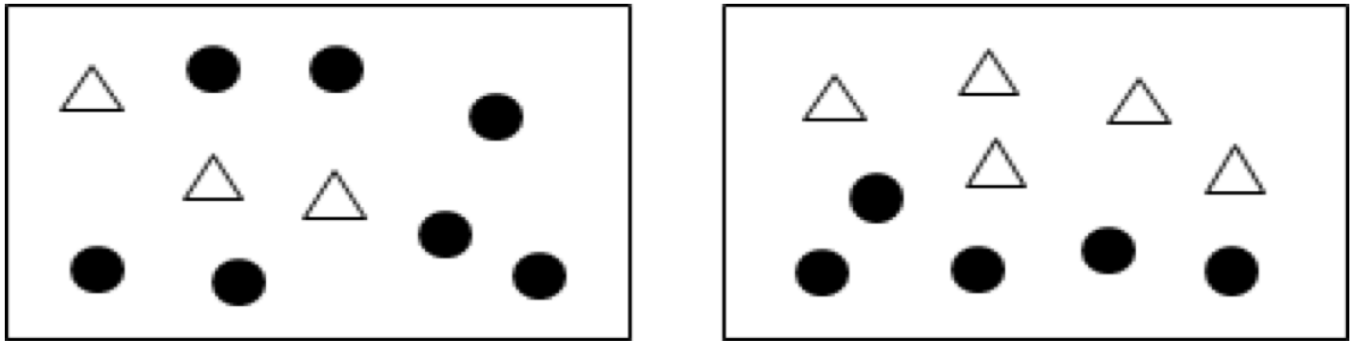
분류

- 판별 기준
  - 결정 트리는 나누어지는 그룹의 순도가 가장 높아지도록 그룹을 나누어야 함
  - 그룹의 순도를 표현하는 데 지니(Gini) 계수 또는 엔트로피(entrophy)가 주로 사용됨
  - Gini 계수는 다음과 같이 정의함

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

분류

- 판별 기준



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

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

분류

- 엔트로피
  - 엔트로피(entropy)도 지니와 유사하게 순도를 나타낸데 사용됨
  - 한 노드(그룹)에 여러 클래스가 골고루 균일하게 섞여 있을 때는 엔트로피가 가장 높고, 동종의 클래스로 모여 있을수록 엔트로피가 낮음(동종의 클래스가 모여있을 수록 좋은 것)
  - 엔트로피의 정의는 아래와 같으며 위와 같은 분류시의 엔트로피를 구하면 아래와 같음

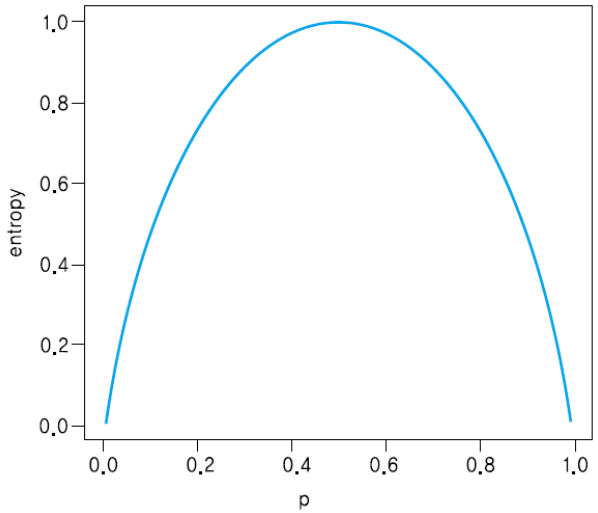
$$Entropy = -\sum_{k=1}^{m} p_k \log_2(p_k)$$

$$\text{엔트로피}(7:3) = -\left[0.7*\log_2(0.7)+0.3*\log_2(0.3)\right] = -\left[(-0.36)+(-0.52)\right] = -(-0.88) = 0.88$$

$$\text{엔트로피}(5:5) = -\left[0.5*\log_2(0.5)+0.5*\log_2(0.5)\right] = -\left[(-0.5)+(-0.5)\right] = -(-1) = 1$$

분류

- 정보량
  - 데이터(이벤트)가 포함하고 있는 정보의 총 기대치, 정보의 가치
  - 정보량을 표현하기 위해 해당 사건이 발생할 확률(probability)을 사용
  - 사건이 발생 확률이 1이라면 정보가 주는 가치가 없고, 사건 발생 확률이 낮을수록 정보가 주는 가치가 높음
  - 정보량은 일어날 확률의 역수에 비례
  - 정보량 = $log\left(\frac{1}{p}\right)$
  - 정보량의 기대치란 어떤 사건이 갖는 가치와 그 사건이 발생할 확률의 곱 - 이를 엔트로피(entropy)라고 함
  - 엔트로피(정보량의 기대치) = p log(1/p) = -p log(p)



분류

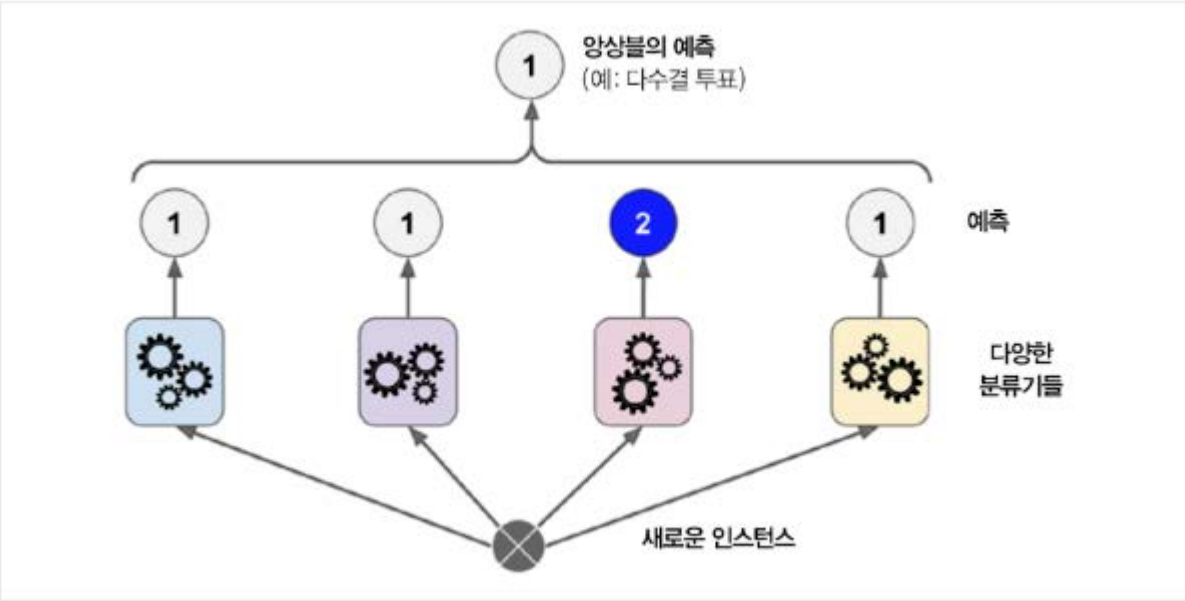
- 종료 조건
  - 결정 트리를 계속 만들어 상세하게 분류를 하면 언젠가는 훈련 데이터에 대해서 100% 순도의 분류가 가능함(전체 엔트로피가 줄어들도록 학습=정보획득량을 최대화할 수 있는 순서대로 속성들을 배치)
  - 이는 과대 적합된 것이므로 테스트 데이터에 대해서는 성능이 오히려 떨어지게 됨
  - 결정 트리 모델은 트리를 만드는 깊이를 제한하지 않으면 과대적합할 위험이 높으므로 주의해야 함
  - 한편 트리의 깊이를 적절한 값보다 너무 작게 제한하면 과소적합이 됨
- 하이퍼파라미터
  - max_depth: 트리의 최대 깊이 (이보다 깊은 트리를 만들지 않는다)
  - max_leaf_nodes: 리프 노드의 최대 수 (리프 노드를 이보다 많이 만들지 않음)
  - min_samples_split: 분할하기 위한 최소 샘플수 (이보다 작으면 분할하지 않음)
  - min_samples_leaf: 리프 노드에 포함될 최소 샘플수 (이보다 작은 노드는 만들지 않음)
  - max_features: 최대 특성수 (분할할 때 이보다 적은 수의 특성만 사용)

분류

- 결정 트리의 특징
  - 결정 트리는 거의 모든 종류의 분류에서 사용할 수 있는 범용 모델이다. 결정 트리의 가장 큰 장점은 알고리즘의 동작을 쉽게 남에게 설명할 수 있다는 것
  - 훈련데이터가 바뀌면 모델의 구조가 달라지는 단점이 있음
  - 결정 트리의 또 다른 장점은 특성 변수의 스케일링이 필요 없다는 것이다. 트리 모델에서는 변수간의 연산이 없기 때문임
    - 각 노드에서는 한 번에 한 특성을 검토하여 어떤 기준으로 트리를 나누면 순도가 올라가는지만 점검하면 되므로 다른 특성 값과의 관계를 계산할 필요가 없음

분류

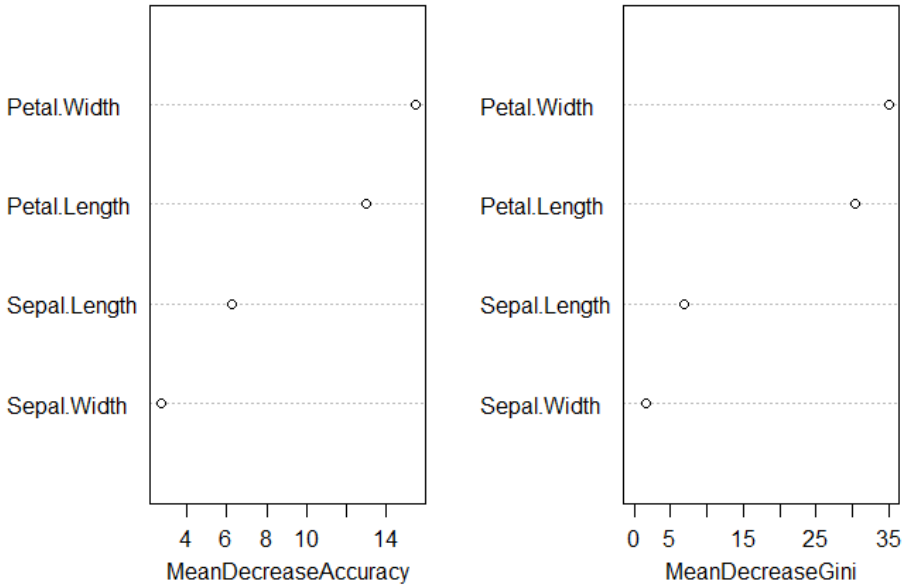
- 랜덤 포레스트
  - 비교적 간단한 구조의 결정 트리 들을 수십~수백개를 랜덤하게 만들고 각 결정 트리의 동작 결과의 평균치를 구하는 방법
  - 이렇게 여러 개의 모델을 만들고 평균을 구하는 방식을 앙상블(ensemble) 방법이라고 하며 하나의 모델만 만드는 것보다 좋은 성능을 보임
  - 주어진 훈련 데이터를 모두 한 번에 사용해서 하나의 최상의 트리 모델을 만드는 방식이 아니라, 데이터의 일부 또는 속성의 일부만 랜덤하게 채택하여 결정 트리를 다양하게 만들고 그 결과의 평균치를 취하는 방식임
  - 나무(tree)가 많이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



https://hoony-gunputer.tistory.com/entry핸즈온-머신러닝-7강-앙상블-학습과-랜덤-포레스트ensemble-RandomForest

분류

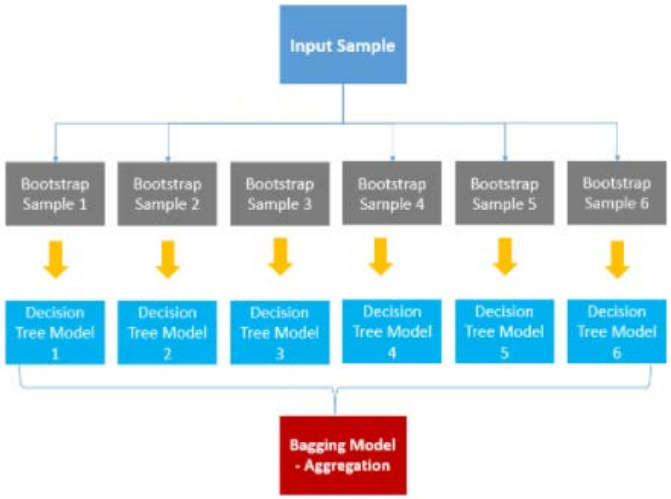
- 랜덤 포레스트의 특징
  - 샘플도 랜덤하게 선택하고, 속성도 랜덤하게 선택하여 다수의 결정 트리를 만들고 이의 평균을 구하면 단일 결정 트리를 사용하는 것보다 안정적이고 우수한 성능을 냄
  - 성능이 우수한 하나의 모델을 사용하는 것보다, 각각의 성능이 최상이 아니지만 다수의 모델을 사용하고 평균치를 구하는 방식이 더 우수함
  - 이를 대중의 지혜, 큰 수의 법칙 등으로 설명하기도 함
  - 랜덤 포레스트의 단점은 모델의 동작을 한가지 트리를 선택하여 설명하기가 어렵다는 것
  - 또한, 계산량이 많아짐

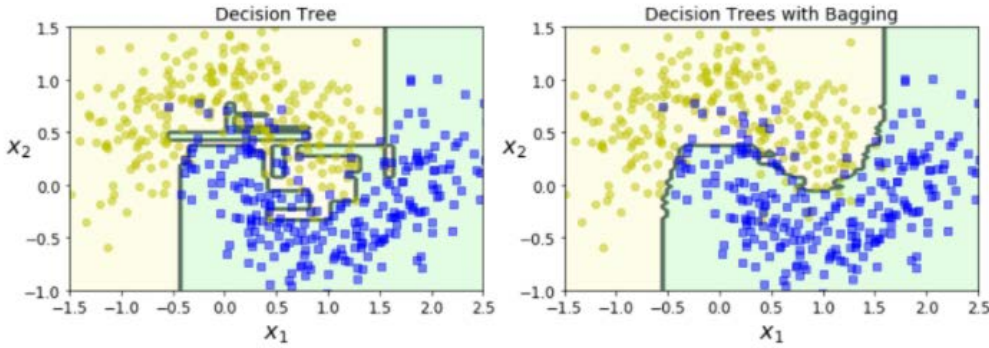



```
#MeanDecreaseAccuracy: 정확도
#MeanDecreaseGini: 노드 불순도 개선
```

분류

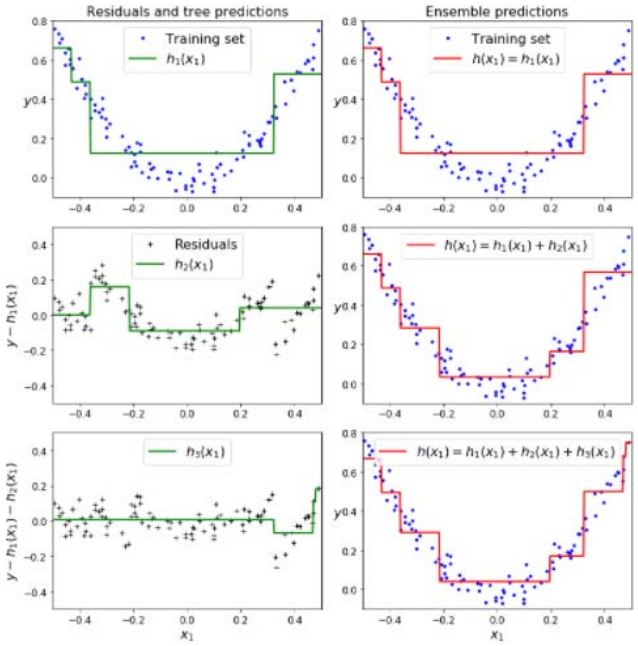
- 배깅
  - 배깅이란 bootstrap aggregation의 줄임말이며 전체 훈련 데이터에서 "중복을 허용"하여 데이터를 샘플링을 하는 방법
    - 중복을 허용하므로 같은 데이터가 중복되어 선택될 수 있음
    - Bootstrap resampling의 줄임말로 부트스트래핑이라고도 부름
- 배깅과 달리 주어진 원래 데이터에서 중복을 허용하지 않고, 즉, 한 번 샘플링 된 것은 다음 샘플링에서 제외하는 방식은 페이스팅(pasting)이라고 함
  - 배깅을 수행하면 학습에 선택되지 않는 샘플은 평균 37%가 되는데 이 샘플을 oob(out of bag) 샘플이라고 한다. 이 oob 데이터는 훈련에 사용되지 않았으므로 검증에 사용하기에 좋다
    - 결정 트리 구조에 배깅을 적용한 방식이 랜덤 포레스트 모델임

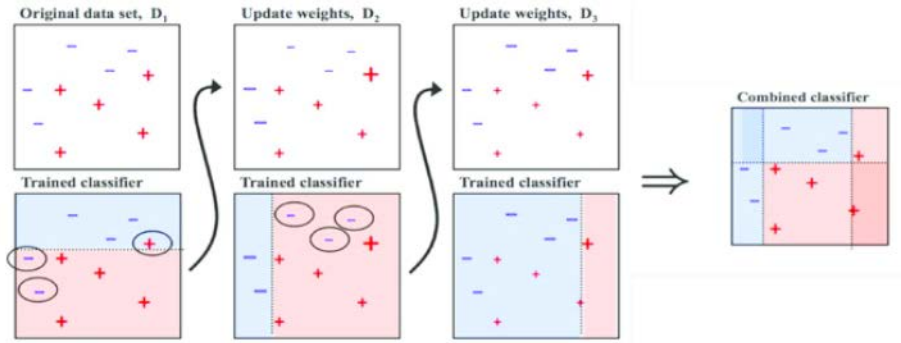




분류

- 그라디언트 부스팅
  - 앙상블 방법 중에 부스팅 알고리즘이 있음
  - 랜덤 포레스트와 달리, 간단한 결정 트리를 다수 만들어 각각 독립적으로 실행한 후에 이들을 평균하는 것이 아니라, 앞의 모델을 보고 성능을 점차 개선하는 방식으로 동작
  - 부스팅에는 아다부스트와 그라디언트 부스트가 널리 사용됨
  - 아다부스트(adaptive boosting)에서는 앞에서 사용한 세부 모델에서 과소적합했던 샘플, 즉 분류에 실패한 샘플의 가중치를 높여주는 것임
  - 즉, 소외되었던 샘플을 주목하여 학습을 다시 시키는 방식이라고 보면 됨





분류

- 분류 모델의 성능
  - 컨퓨전 매트릭스란 분류의 결과가 잘 맞았는지를 평가하는 채점표와 유사
  - 결과 값이 P(Positive)또는 N(Negative) 둘 중 하나만 가질 수 있는 binary 예측의 경우를 설명하는 일반적인 용어
  - Positive는 찾고자 하는 현상(ex. 암에 걸린 사실, 결함 등)이 나타난 것인지를 구분하는 것일 뿐, 긍정적인 결과를 찾았다는 뜻은 아님

실제 \ 예측	P로 예측	N로 예측
실제로 P	True positive (TP)	False negative (FN)
실제로 N	False positive (FP)	True negative (TN)

분류

- 용어의 의미 예시
  - True positive (TP)
    - 암/결함이라고 예측했는데 실제로 암에 걸린 경우
  - False positive (FP)
    - 암/결함이라고 예측했는데 실제는 암에 걸리지 않은 경우
  - False negative (FN)
    - 암/결함이 아니라고 예측했는데 실제는 암인 경우
  - True negative (TN)
    - 암/결함이 아니라고 예측했는데 실제로도 암이 아닌 경우

첫 번째 단어: 예측 평가	두 번째 단어: 추정 내용
True: 예측이 맞음	Positive: positive로 예측
False: 예측이 틀림	Negative: negative로 예측

분류

- 평가 지표
  - 정확도(accuracy): 정확하게 예측한 비율을 의미
    - accuracy = (TP+TN) / 전체 경우의 수(N)

실제 \ 예측	암이라고 예측	암이 아니라고 예측	합계
실제 암환자	6 (TP)	4 (FN)	10
실제로 암환자 아님	2 (FP)	188 (TN)	190
합계	8	192	200

  - 암진단 정확도 = (6 + 188)/200 = 194/200 = 0.97 => 97%
  - 오류율 = 1-accuracy = 0.03 => 오진율은 3%
  - 리콜(recall): 관심 대상을 얼마나 잘 찾아내는가
    - recall = TP / (TP+FN)
    - 실제 암 환자 발견률 = 6 / (6+4) = 0.6 => 60%
  - 정밀도(precision): 예측의 정확도
    - precision = TP / (TP+FP) = 6 / (6+2) = 0.75 => 75%

분류

- F1_score
  - recall과 precision의 두 가지 지표를 동시에 높이는 것은 어려움, F1은 이러한 두 요소를 동시에 반영한 새로운 지표임
  - F1은 recall과 precision의 조화 평균을 구한 것
  - F1 = 2 x precision x recall / (precision + recall)
  - 두 지표의 값이 각각 0.5와 0.7일 때
    - 산술 평균 c=(a+b)/2=(0.5)+(0.7)/2=0.6
    - 조화 평균 c=2ab/(a+b)=0.7/1.2=0.58
  - 두 지표의 값이 각각 0.9와 0.3일 때
    - 산술 평균 c=(a+b)/2=(0.9)+(0.3)/2=0.6
    - 조화 평균 c=2ab/(a+b)=0.54/1.2=0.45

조화 평균: $\frac{1}{c}=\frac{(\frac{1}{a}+\frac{1}{b})}{2}$

$$c=\frac{2ab}{a+b}$$

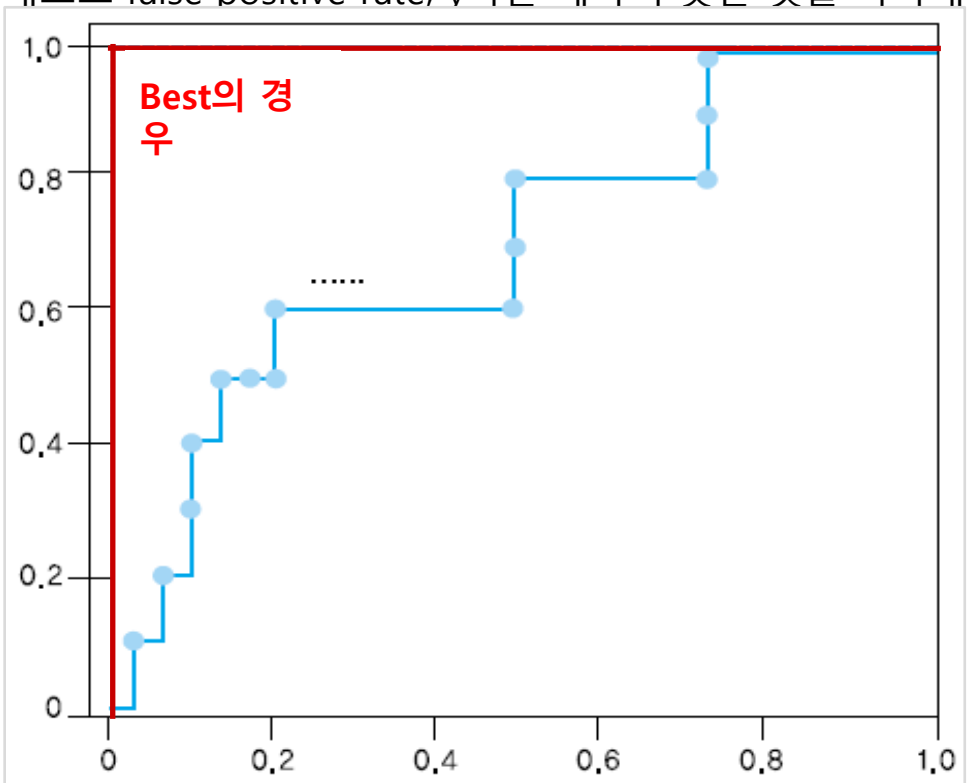
분류

- 분류 순서 평가

환자번호	성별	점수	순위	실제 값
7	F	0.98	1	N
125	M	0.96	2	C
4	F	0.95	3	N
199	M	0.86	4	C
2	F	0.84	5	N
200	M	0.82	6	C
176	M	0.81	7	C
73	M	0.80	8	N
82	M	0.79	9	C
3	F	0.77	10	N
123	F	0.76	11	N
		…		C
43	F	0.48	198	N
93	M	0.42	199	N
120	F	0.40	200	N

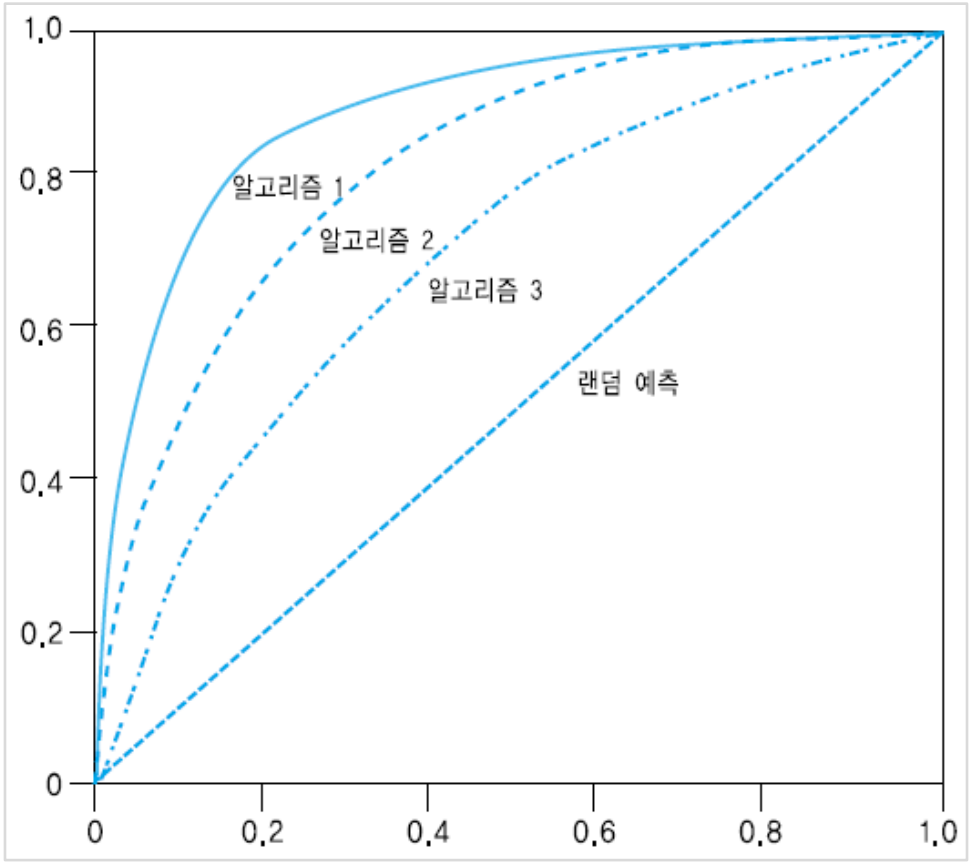
분류

- ROC(Receiver Operating Characteristic)
  - 예측 결과를 순서대로 제시한 것이 실제 값과 얼마나 순서에 따라 잘 맞는지는 검증하는 2차원 그래프
  - ROC 커브는 (0,0)점에서 시작하여 한 행씩 진행하면서 정답을 맞추었으면 y축 위로 한 칸 이동, 정답을 맞추지 못했으면 x축 방향으로 한 칸 이동. 종점은 (1, 1) 지점
  - 그래프의 x 축으로는 예측 오류가 날 때마다 이동하고, y축으로는 정답을 맞출 때마다 이동
  - x축은 예측이 틀린 것을 나타내므로 false positive rate, y축은 예측이 맞은 것을 나타내므로 true positive rate를 나타냄



분류

- AUC(Area Under Curve)
  - 예측 알고리즘의 성능을 간단히 수치로 나타내기 위해서 ROC 그래프의 면적을 계산하는 방법을 사용
  - 우수한 알고리즘일수록 초반에 y축 상단 방향으로 이동하므로 ROC 커브의 면적이 넓어짐



하이퍼파라미터 최적화

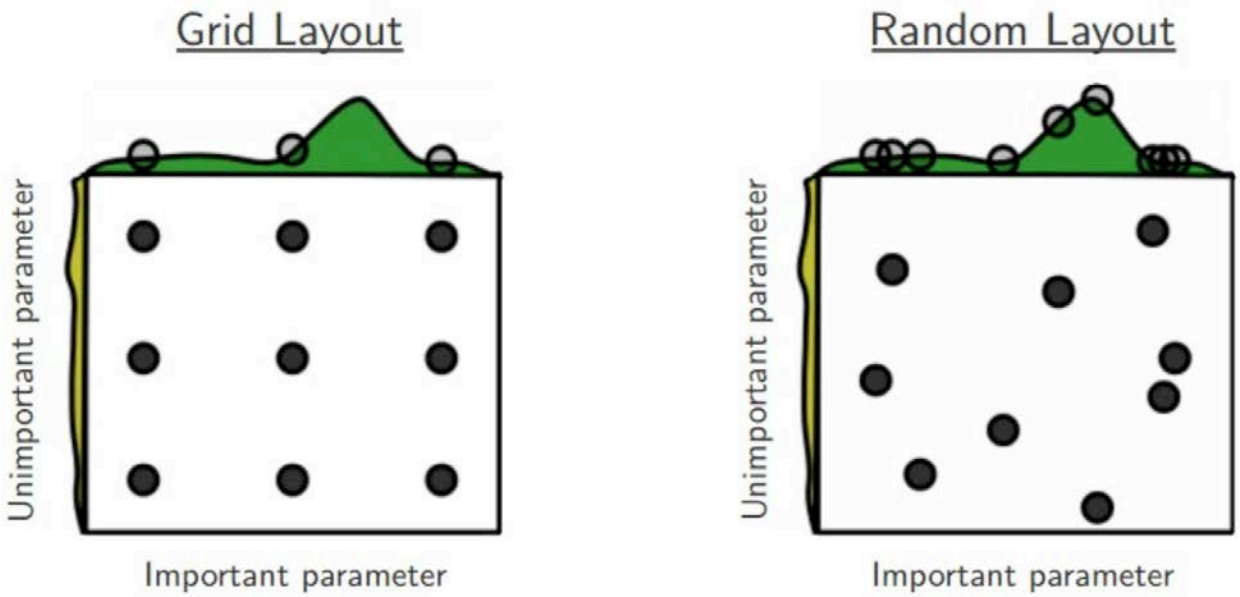
- 최적의 머신러닝 모델을 만드는 과정은 모델을 설계하고, 학습하고, 하이퍼파라미터 최적화 세 단계로 구성됨
  - 모델 구조(알고리즘) 선택
  - 파라미터 학습
  - 모델 최적화 : 하이퍼 파라미터 선택
- 최적의 성능을 내는 모델을 완성하려면 모델의 구조를 구성하는 하이퍼파라미터의 최적값을 찾아야 함
- 이 과정에서 과대적합과 과소적합을 피하면서 성능평가지표를 최대로하는 하이퍼 파라미터를 찾음

하이퍼파라미터 최적화

- 그리드 탐색
  - 하이퍼 파라미터가 가질 수 있는 전체 범위를 몇 개의 구간으로 나누어 일일이 하나씩 점검해보는 방식
    - 예를 들어, SVC 모델을 사용할 때 gamma 변수와 C 변수의 값을 각각 5가지, 4가지로 나누고 총 4x5=20가지 경우를 시도한다.
    - 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, 트리깊이, 분할할 조건, 분류할 최소 샘플 수

하이퍼파라미터 최적화

- 랜덤 탐색
  - 그리드 탐색은 단순하나 여러 경우의 수를 모두 탐색하는데 시간도 오래 걸리고 최적의 값을 놓치는 경우가 있음
    - 격자 모양의 파라미터 조합의 값 사이에 실제로 최적의 하이퍼파라미터 값이 있었다면 이를 찾아낼 방법이 없음
  - 랜덤 탐색은 두 단계로 수행됨(RandomizedSearchCV()함수 사용)
    - 일정한 범위 내에서 랜덤하게 하이퍼파라미터를 선택하여 성능을 실험하여 대체로 어떤 영역에서 성능이 좋은지를 찾음
    - 다음에는 이 영역을 중심으로 세밀하게 탐색

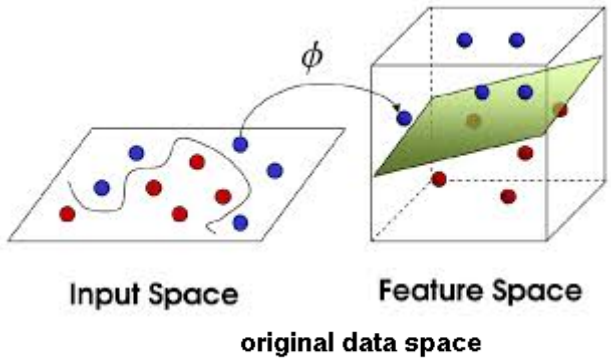


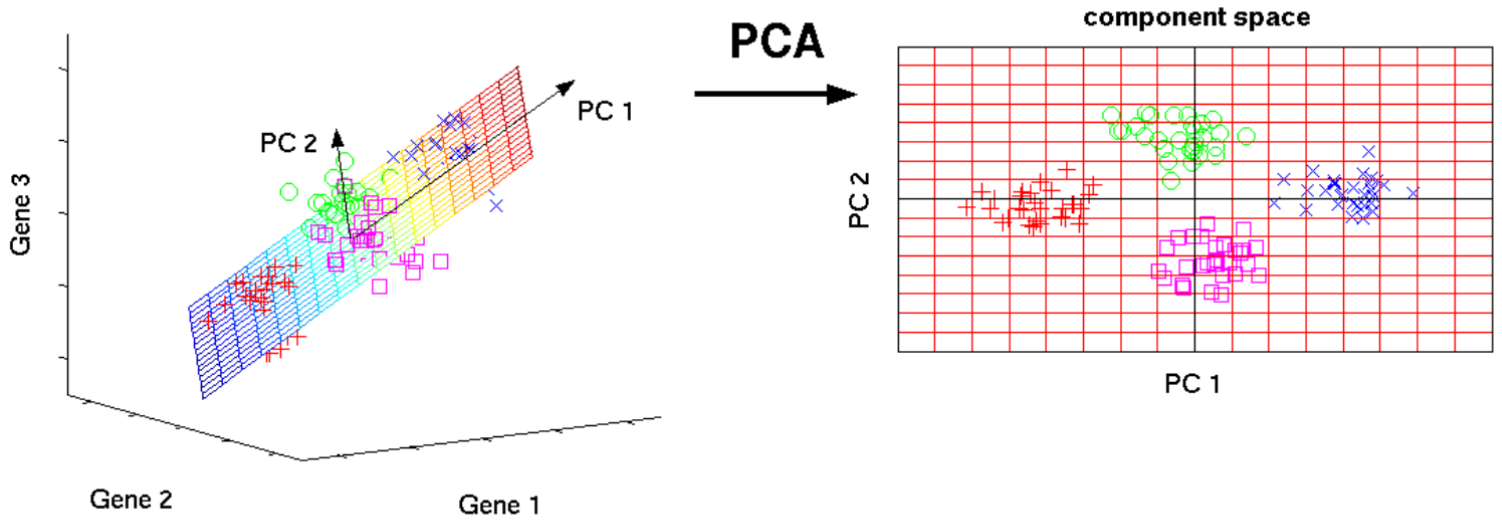
차원 축소

- 차원 축소
  - MNIST 이미지를 다시 보면..
    - 이미지의 대부분은 흰색이므로 훈련 데이터셋에서 이런 픽셀을 제거해도 많은 정보를 잃지 않음
    - 인접한 두 픽셀은 종종 연관성이 높기때문에, 두 픽셀을 하나로 합쳐도 많은 정보를 잃지 않음

차원 축소

- 차원 축소
  - 입력 데이터의 차원의 저주를 피하기 위해 차원을 축소하는 방법

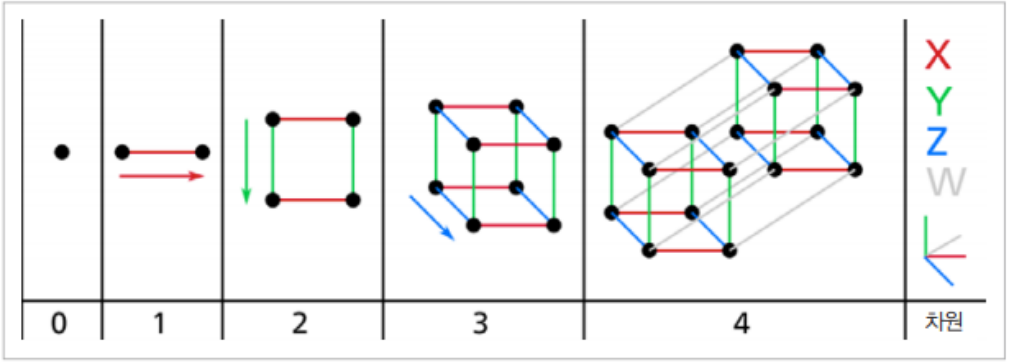




Reduce unnecessary representation axis

차원 축소

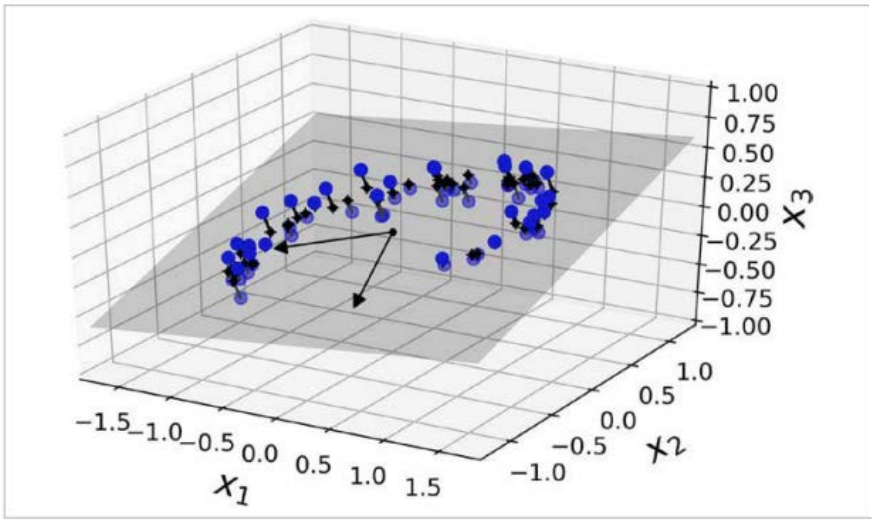
- 차원의 저주
  - 우리는 3차원의 세계에서 살고 있음(4차원 이상으로만 가도 직관적 이해 불가능)
  - 2차원 단위 면적에서 임의의 두 점을 선택했을 때 두 점 사이의 거리는 평균적으로 0.52인데, 1000000차원에서는 이것이 408.25까지 늘어남
  - 이론적인 해결법은 고차원에서도 데이터끼리의 거리가 가까울 수 있도록 즉 밀도가 높아질 때까지 dataset의 크기를 키우는 것(차원 수가 커질수록 필요한 데이터 수가 기하급수적으로 커지기에 현실적으로 불가능)

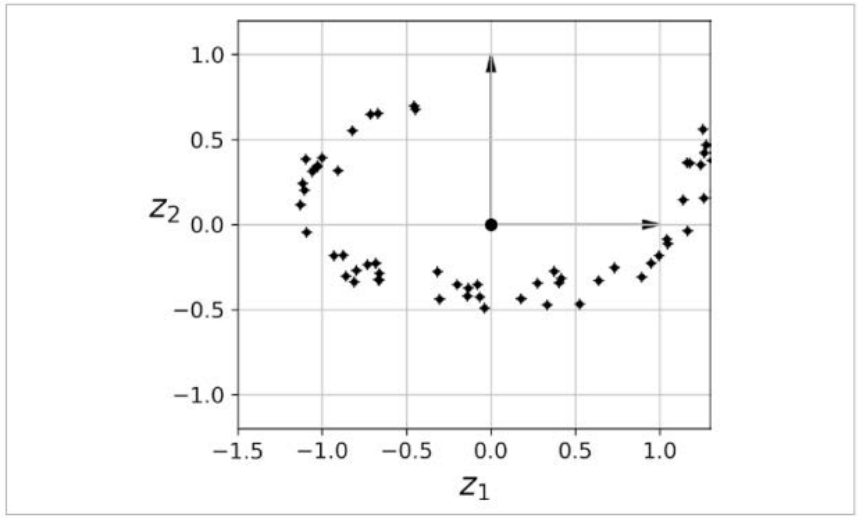


https://box-world.tistory.com/61

차원 축소

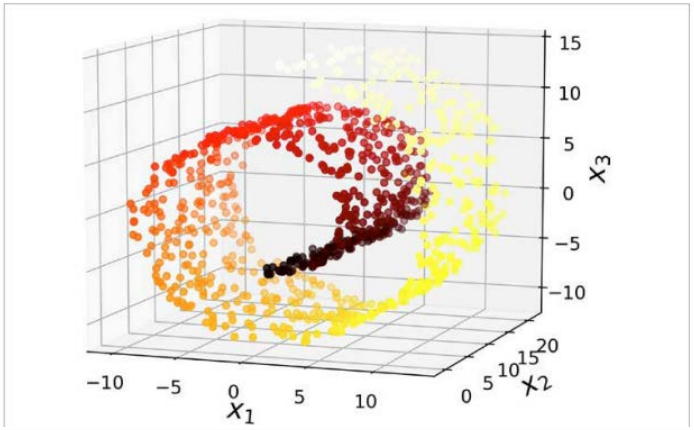
- 차원축소를 위한 접근법
  - 투영
    - 데이터셋은 모든 차원에 균일하게 퍼져있지 않고 특정 피쳐들끼리 연관을 갖는 경우가 많음
    - 데이터가 고차원 공간안에서 저차원에 놓여있음(특정 피쳐들끼리 연관성을 가짐

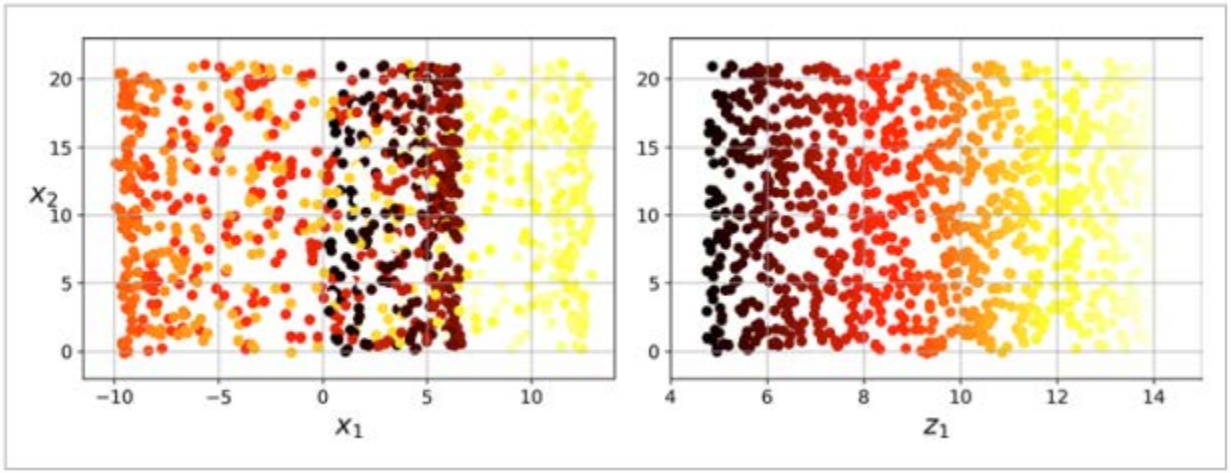




차원 축소

- 차원축소를 위한 접근법
  - 투영
    - 스위스 롤 데이터셋에서는 투영이 좋은 방법이 아님

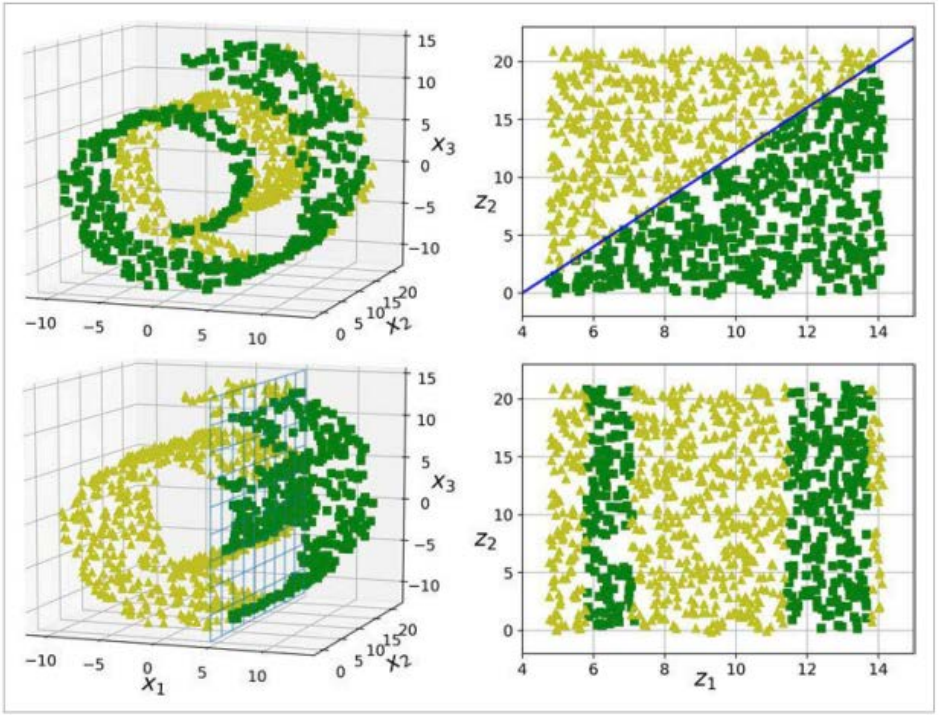




https://box-world.tistory.com/61

차원 축소

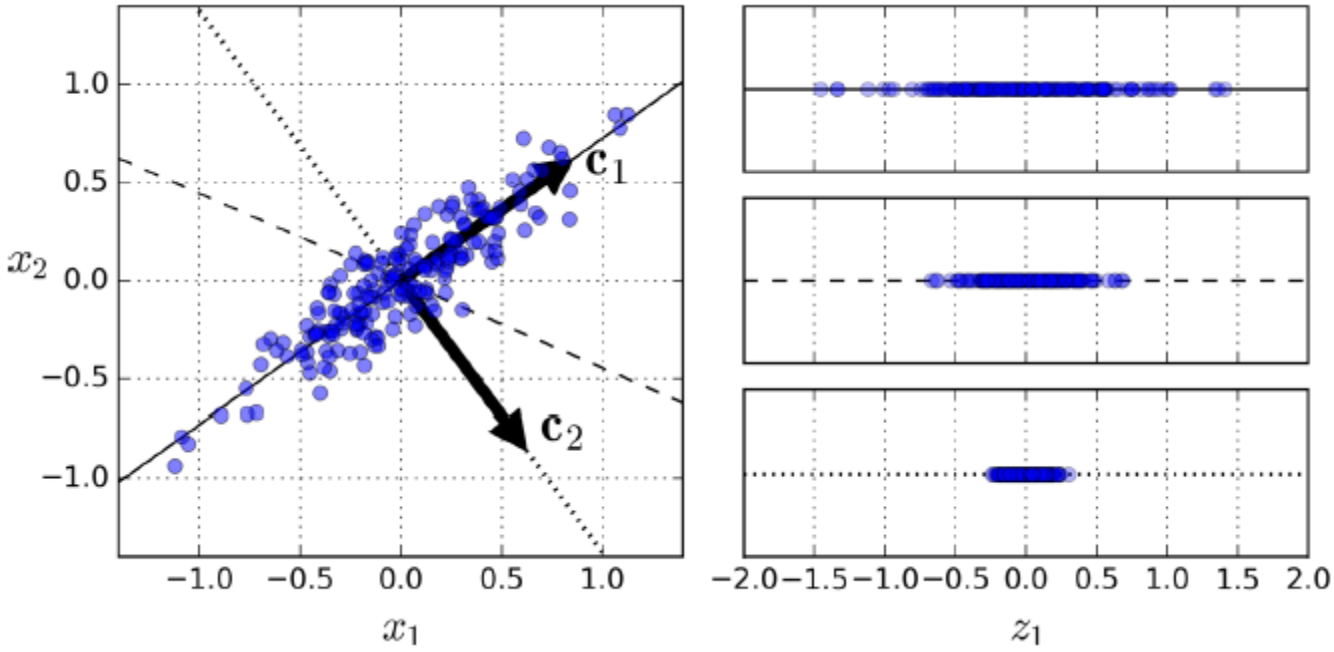
- 차원축소를 위한 접근법
  - 매니폴드 학습
    - 스위스 롤 예제는 2D 매니폴드의 한 예시임(고차원 공간에서 휘어지거나 뒤틀린 2D 모양)
    - 차원축소 알고리즘은 훈련 샘플이 놓여 있는 매니폴드를 모델링하는 식으로 작동하는 것을 매니폴드 학습이라고 함(꼬여있는 매니폴드를 풀어헤침)
    - 분류나 회귀 작업 시 저차원 매니폴드 형태로 데이터를 표현하면 더 간단해질것이라고 가정함



https://box-world.tistory.com/61

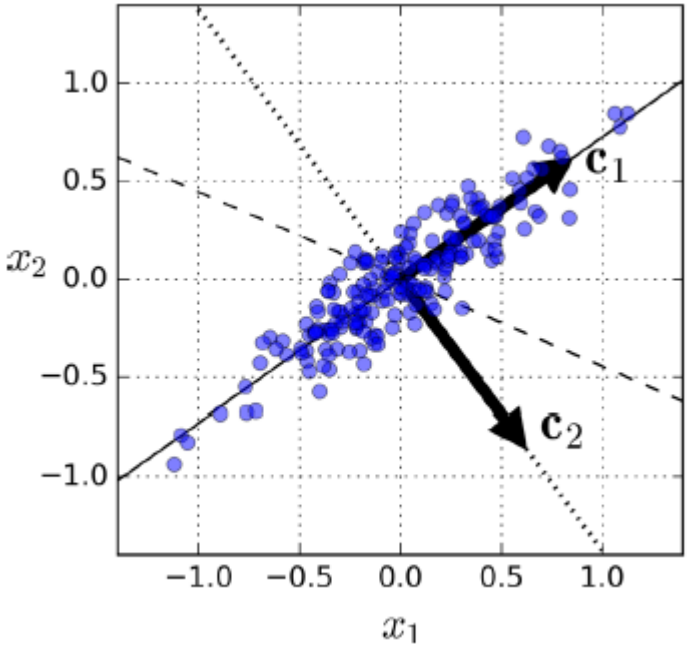
차원 축소

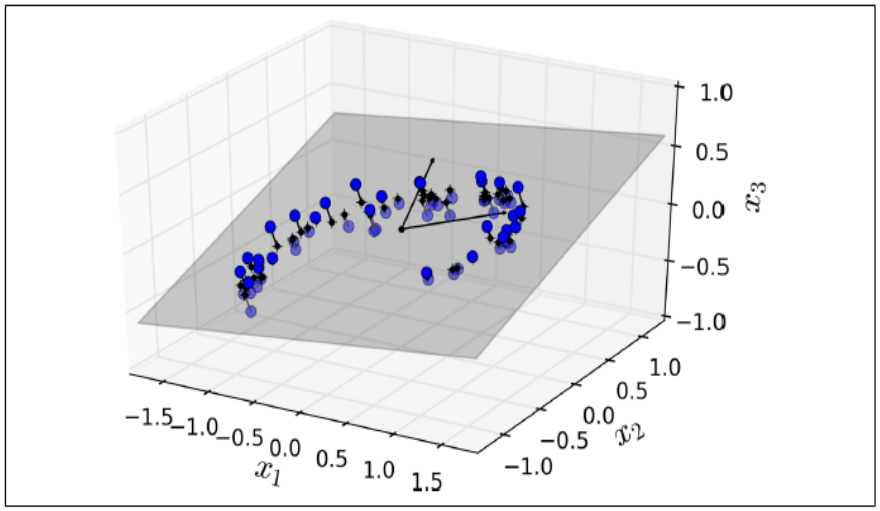
- PCA(주성분 분석)
  - 데이터에 가장 가까운 초평면을 정의한다음, 데이터를 이 평면에 투영시킴
  - 저차원의 초평면에 훈련 세트를 투영하기 전, 올바른 초평면을 선택해야 함
  - 다른 방향으로 투영하는 것보다 **분산이 최대로 보존하는 축을 선택하는 것이 정보가 가장 적게 손실**되므로 합리적으로 보임
    - 원본 데이터셋과 투영된 것 사이의 평균 제곱 거리를 최소화하는 축



차원 축소

- PCA(주성분 분석)
  - PCA는 훈련 세트에서 분산이 최대인 축을 찾음
  - 첫 번째 축에 직교하고 남은 분산을 최대로 보존하는 두 번째 축을 찾음
- 2D 예제에서는 선택의 여지가 없음 => 점선이 됨
  - 고차원 데이터셋이라면 PCA는 이전의 두 축에 직교하는 세 번째 축을 찾으며
  - 데이터셋에 있는 차원의 수만큼 n번째 축을 찾음





https://box-world.tistory.com/61

차원 축소

- PCA(주성분 분석)
  - 특이값 분해
    - 훈련 세트의 주성분을 찾는 방법
    - 표준 행렬 분해 기술로, 훈련 세트 행렬 X를 세 개 행렬의 행렬 곱셈인 $U\Sigma V^T$ 로 분해할 수 있음
    - 찾고자 하는 모든 주성분의 단위 벡터가 V에 다음과 같이 담겨 있음

$$\mathbf{V}^T = \begin{pmatrix} | & | & & | \\ \mathbf{c_1} & \mathbf{c_2} & \cdots & \mathbf{c_n} \\ | & | & & | \end{pmatrix}$$

주성분 행렬

  - PCA는 데이터셋의 평균이 0이라고 가정
  - 사이킷런의 PCA 파이썬 클래스는 이 작업을 대신 처리하지만, PCA는 구현하거나 다른 라이브러리를 사용한다면 먼저 데이터를 원점에 맞추어야 함

```
X_centered = X - X.mean(axis=0)
U, s, Vt = np.linalg.svd(X_centered)
c1 = Vt.T[:, 0]
c2 = Vt.T[:, 1]
```


특이값분해 선형대수학 설명 : https://bskyvision.com/251

차원 축소

- PCA(주성분 분석)
  - 투영하기
    - 주성분을 모두 추출했다면, 처음 d개의 주성분으로 정의한 초평면에 투영하여 데이터셋의 차원을 d차원으로 축소 할 수 있음
    - 초평면에 훈련 세트를 투영하고 d차원으로 축소된 데이터셋 Xd-proj 을 얻기 위해서는 아래와 같이 행렬 X와 V의 첫 d열로 구성된 행렬 Wd를 행렬 곱셈하면 됨

$$\mathbf{X}_{d\text{-proj}} = \mathbf{X} \cdot \mathbf{W}_d$$

```
W2 = Vt.T[:, :2]
X2D = X_centered.dot(W2)
```

  - 사이킷런으로 PCA 분석하기

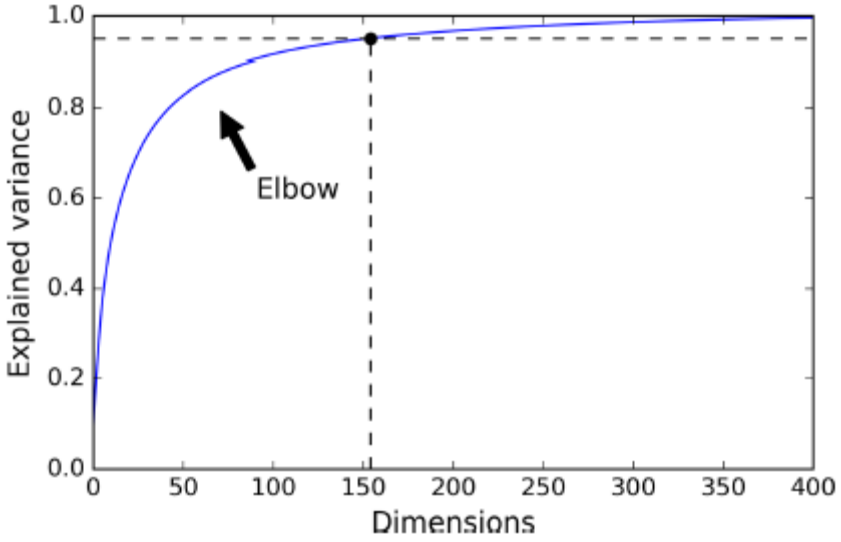
```
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)
X2D = pca.fit_transform(X)
```

    - 첫번째 주성분을 정의하는 단위 벡터 **pca.components_.T[:, 0]**
    - **분산의 비율 pca.explained_variance_ratio_ = array([**0.84248607, 0.14631839**])**

특이값분해 선형대수학 설명 : https://bskyvision.com/251

차원 축소

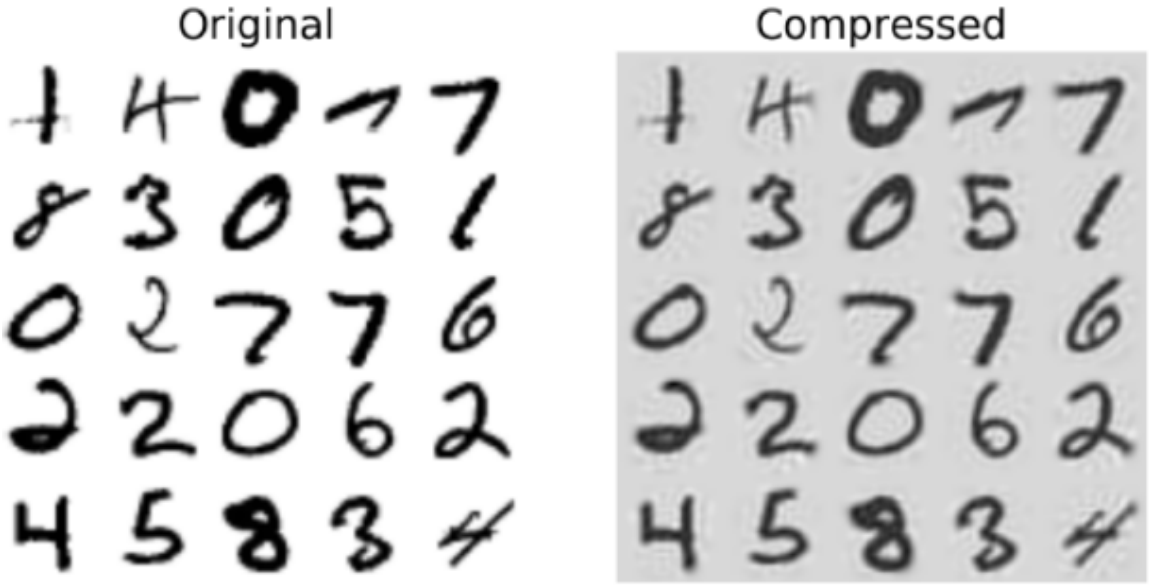
- PCA(주성분 분석)
  - 적절한 차원 수를 선택
    - 축소할 차원의 수를 임의로 정하기 보다는 충분한 분산(95%)이 될때까지 더해야할 차우너 수를 선택하는 것이 좋음



특이값분해 선형대수학 설명 : https://bskyvision.com/251

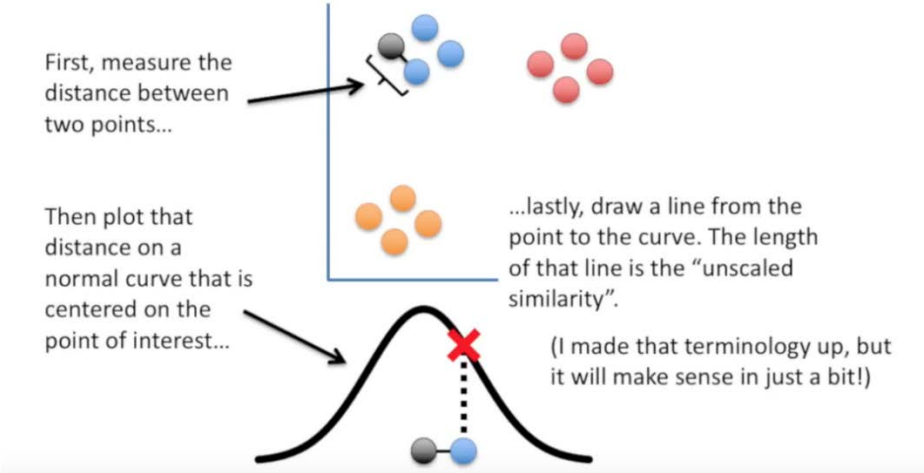
차원 축소

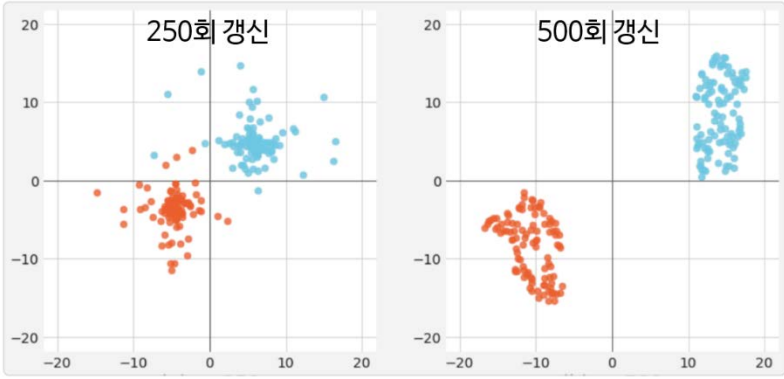
- PCA(주성분 분석)
  - MNIST예시로 돌아가서..
  - MNIST 데이터셋에 분산의 95%를 유지하도록 PCA를 적용하면
    - 784개의 특성이 150개정도로 변환됨
    - 대부분 분산은 유지되었지만 데이터셋은 원본 크기의 20% 미만이 됨
    - 데이터 압축은 분류 알고리즘의 속도를 높일 수 있음
  - 압축된 데이터셋에 PCA를 반대로 적용하여 다시 되돌릴 수 있음
    - 손실된 5%는 유실된 상태로 복구되므로 원본 데이터셋으로 되돌릴 수는 없음



차원 축소

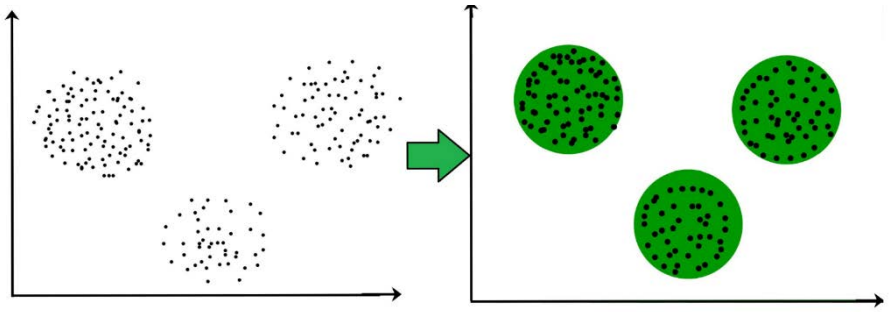
- T-SNE(t-Distributed Stochastic Neighbor Embedding)
  - 비슷한 샘플은 가까이, 비슷하지 않은 샘플은 멀리 떨어지도록 하면서 차원을 축소함
  - 고차원 공간에 있는 샘플의 군집을 시각화하는데 주로 사용됨
- 동작 방식
  - 점을 하나 선택(검정)
  - T 분포를 이용하여 검정(기준)점과 상대점과의 거리(유사도)를 구함(t 분포의 값을 선택)
  - 거리가 가까운 점들끼리 묶어 줌
- 장단점
  - PCA처럼 군집이 중복되지 않는다는 장점이 있음
  - 매번 계산할 때마다 축의 위치가 바뀜(다른 모양이 나타남)
  - 머신러닝 학습 Feature로 사용할 수 없음(매번 값이 바뀌기때문)

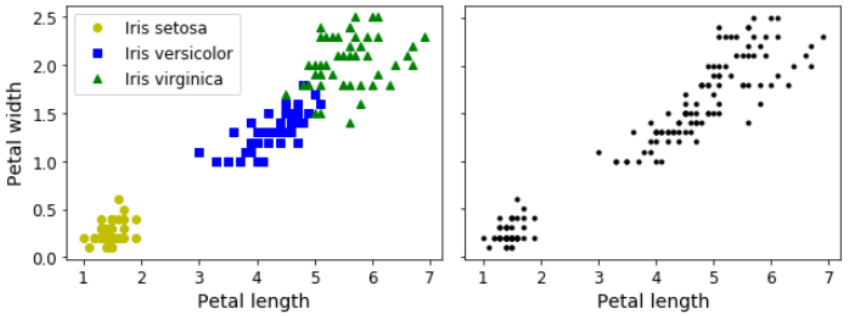




클러스터링

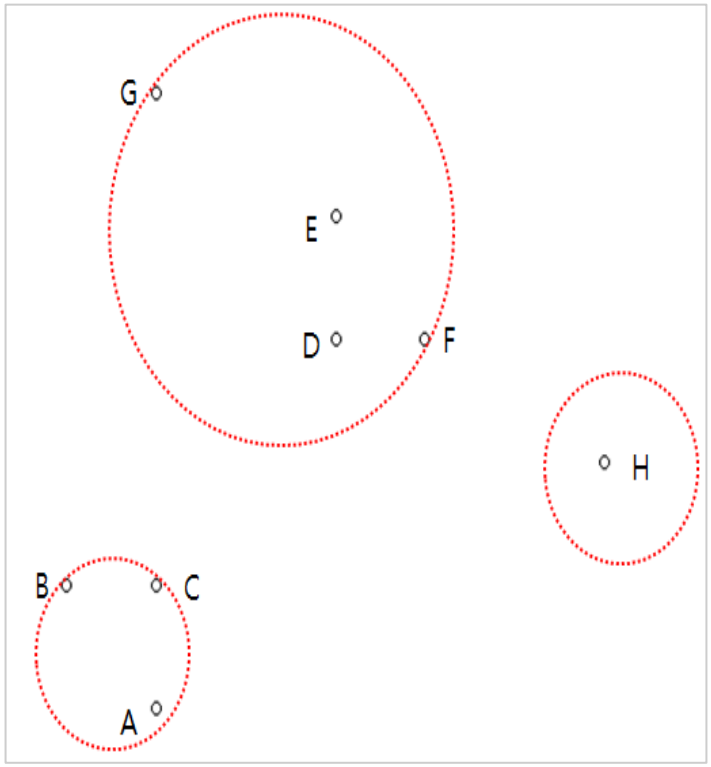
- 클러스터링
  - 비슷한 샘플을 구별해 하나의 클러스터 또는 비슷한 샘플의 그룹으로 할당하는 작업
  - K개의 그룹에 속하는 유사한 샘플을 찾는 과정
  - 응용분야
    - 고객분류 : 구매이력, 웹사이트 행동을 기반으로 클러스터를 모을 수 있음
    - 데이터 분석 : 새로운 데이터가 들어올 때 어떤 군집에 속할지 판단
    - 차원 축소 : 군집알고리즘을 적용하면 샘플의 친화성을 계산할 수 있음(K개의 클러스터 K차원의 친화성 벡터)
    - 이상치 탐지 : 모든 클러스터에 친화성이 낮은 샘플은 이상치로 판단가능
    - 준지도 학습 : 동일한 클러스터에 있는 모든 샘플에 레이블 전파가 가능
    - 검색 엔진 : 이미지 검색과 같은 경우 비슷한 이미지의 클러스터를 구축하고 있어야 함
    - 이미지 분할 : 윤곽 탐지, 물체 감지

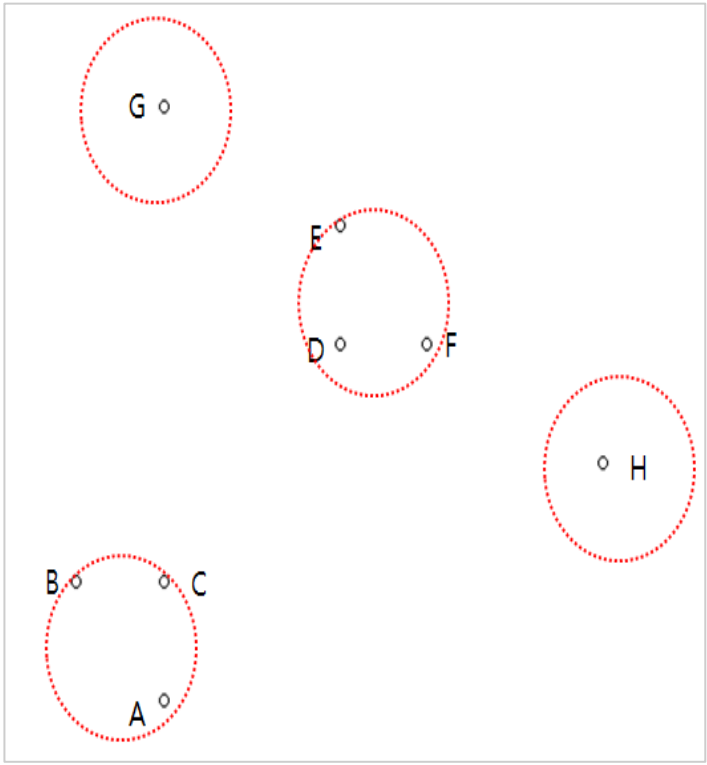




클러스터링

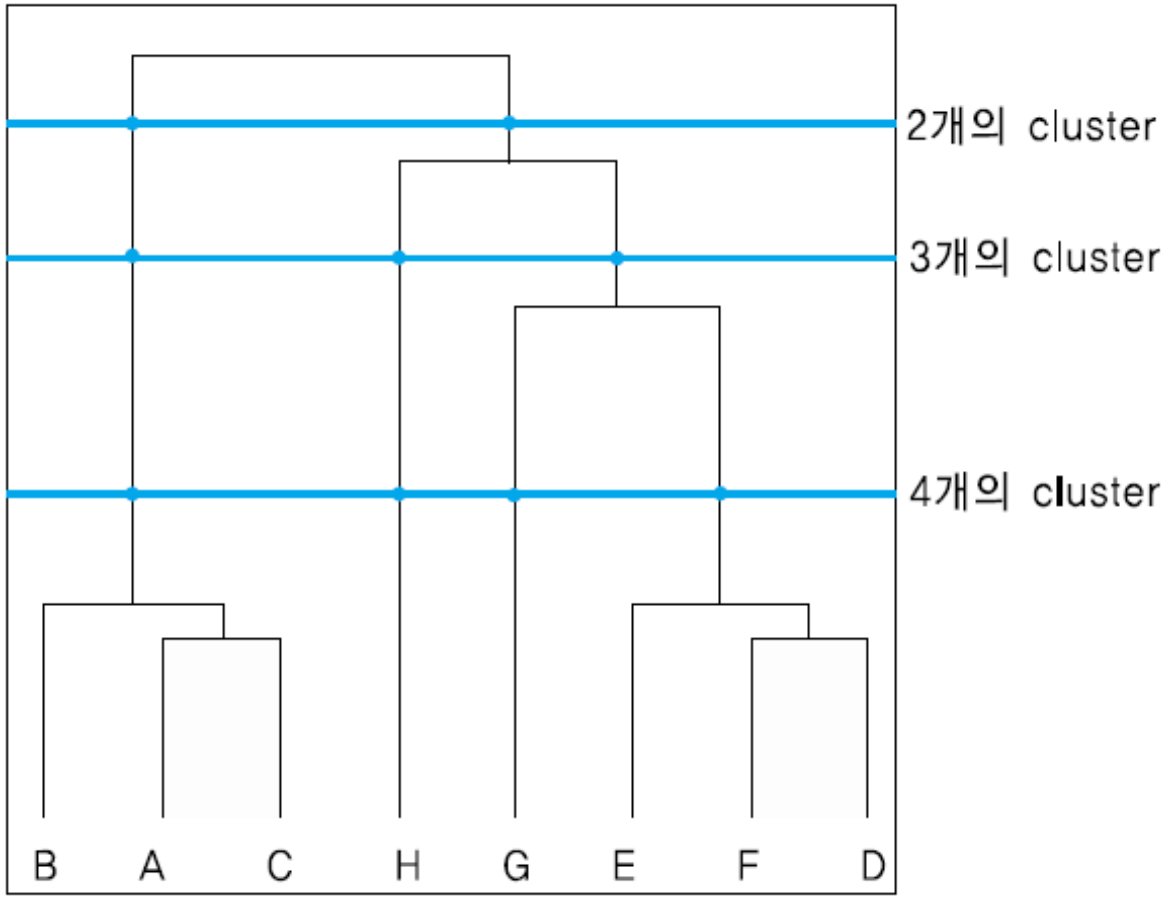
- 클러스터링
  - 적정한 군집의 수(k)를 먼저 찾아야 함





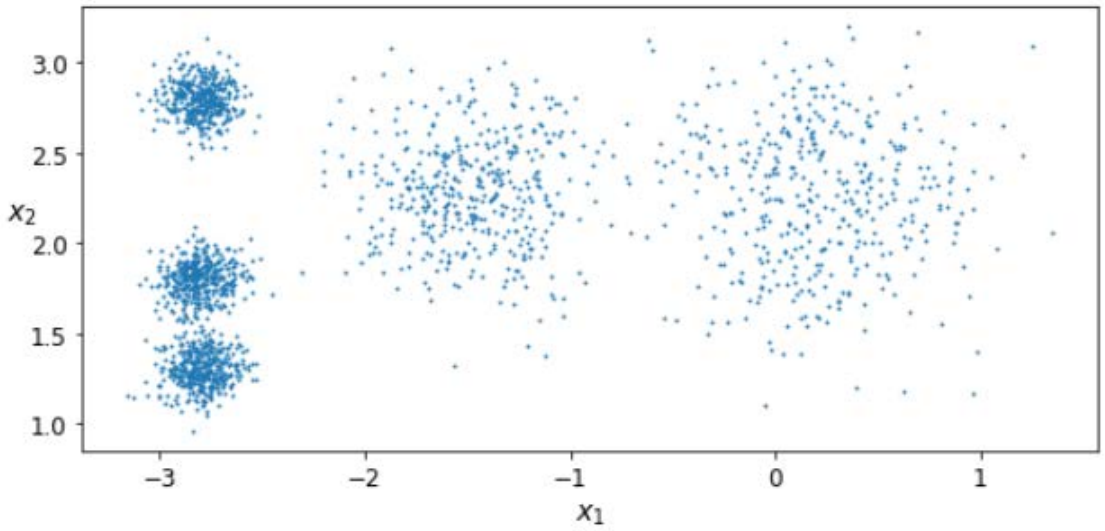
클러스터링

- 클러스터링
  - 적정한 군집의 수(k)를 먼저 찾아야 함



클러스터링

- 클러스터링 알고리즘
  - 조건
    - 같은 그룹 내의 항목들은 서로 속성이 비슷함 (유사도가 큼)
    - 다른 그룹에 속한 항목과는 속성이 서로 다름 (유사도가 작음)
  - 비정상 패턴 (이상치) 식별에도 사용됨
- K-means 알고리즘
  - 데이터 샘플 중 덩어리로 잘 묶어지는 종류의 데이터 샘플을 빠르게 클러스터로 묶을 수 있는 알고리즘으로 **각 클러스터의 중심을 찾고 가장 가까운 클러스터에 샘플을 할당**



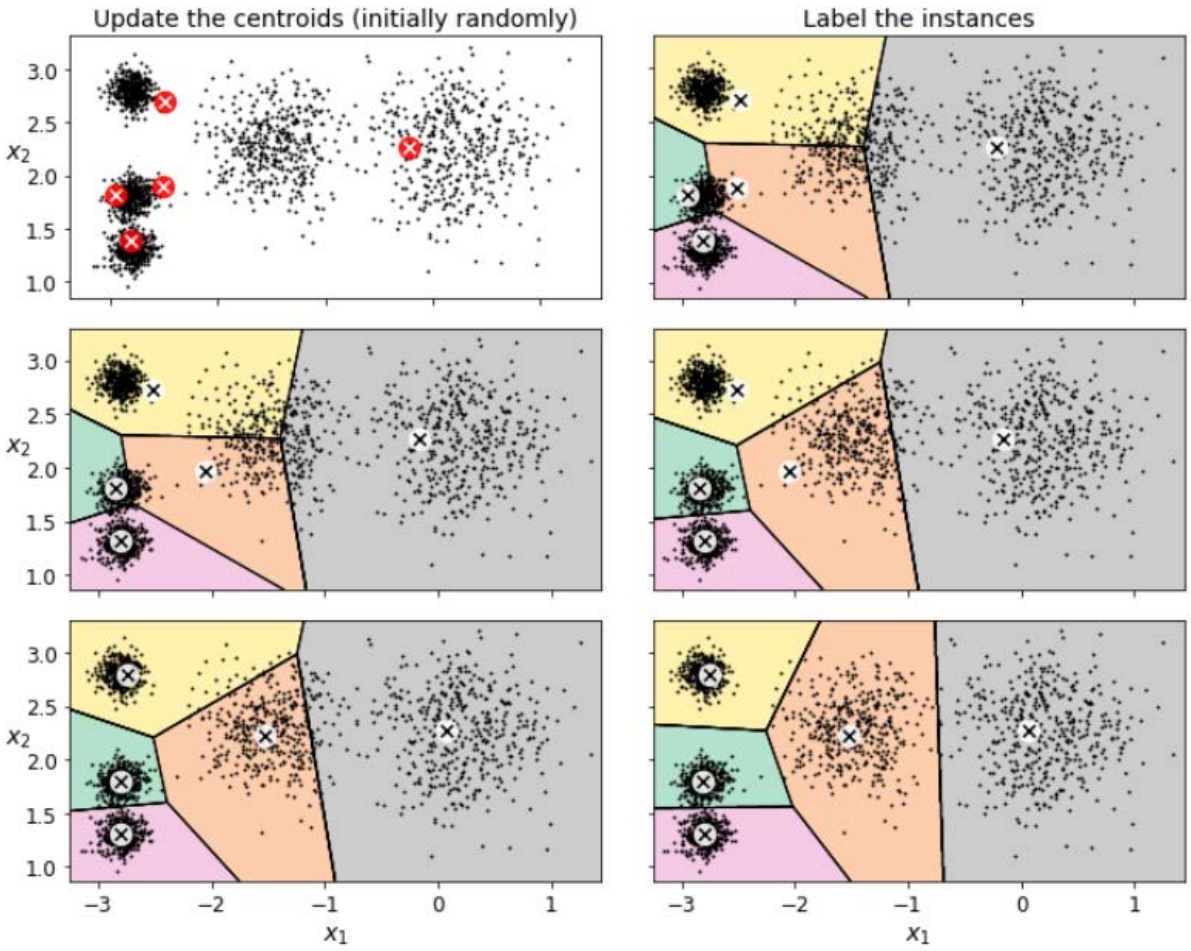
샘플 덩어리 다섯 개로 이루어진 레이블 없는 데이터셋

클러스터링

- K-means 알고리즘
  - 공간상에 임의의 k 개의 임의의 초기 지점을 클러스터 중점으로(cluster center) 정함
  - 클러스터 중점을 중심으로 거리가 가까운 항목을 선택하여 클러스터 공간을 나눔
  - 각 클러스터에 포함된 항목들의 평균 위치를 구해 이를 새로운 클러스터 중점(centroid)으로 변경
  - 새로 설정된 센트로이드를 중심으로 경계를 다시 그림
    - 각 항목들이 소속된 클러스터가 바뀔 수 있음
  - 변경된 항목들을 가지고 클러스터 중심을 다시 계산
  - 더 이상 클러스터의 모양이 바뀌지 않을 때까지 반복 수행함
    - KMeans() 사용

클러스터링

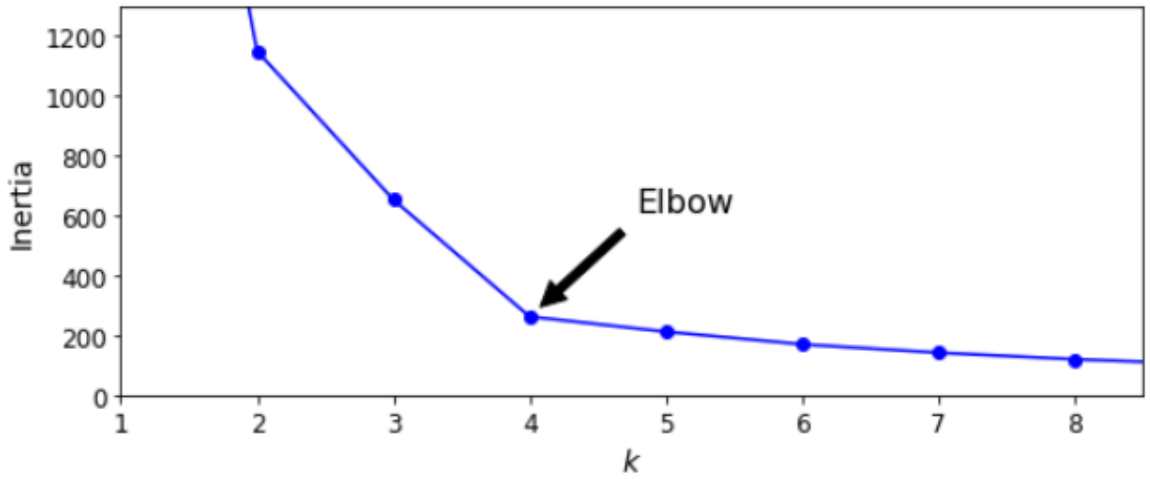
- K-means 알고리즘



센트로이드 업데이트 그래프 / 샘플 레이블 할당 그래프

클러스터링

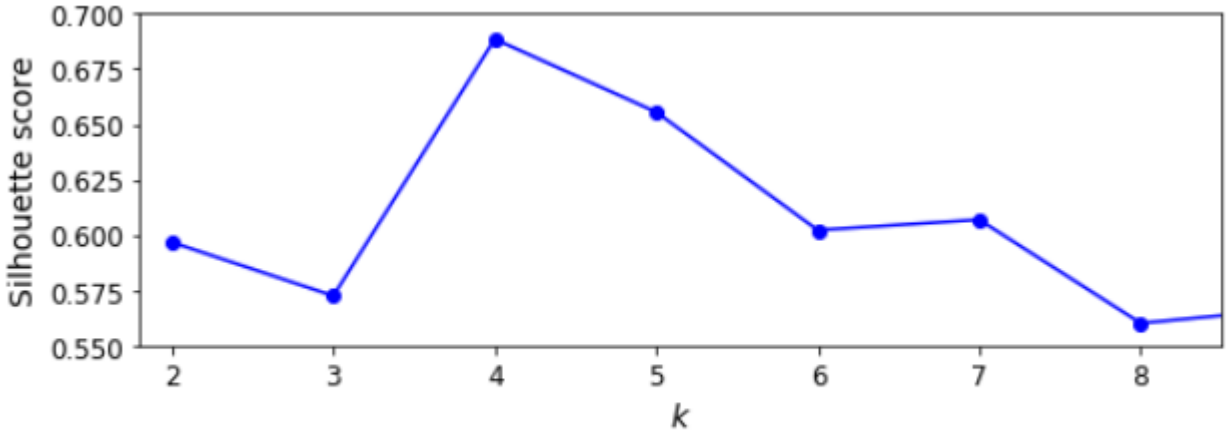
- K-means 알고리즘
  - 최적의 클러스터 수 K 결정하기
    - 클러스터 개수를 올바르게 지정하지 않으면 결과가 매우 나빠질 수 있음
    - 모델의 이너셔(Inertia)를 사용
    - 이너셔 : 각 샘플과 가장 가까운 센트로이드 사이의 평균 제곱거리
    - 이너셔는 K값이 증가함에 따라 점점작아지므로 k값을 선택할 때 좋은 성능 지표가 아님



이너셔 그래프를 클러스터 개수 k의 함수로 그렸을 때 그래프가 꺽이는 지점을 엘보라고 부름

클러스터링

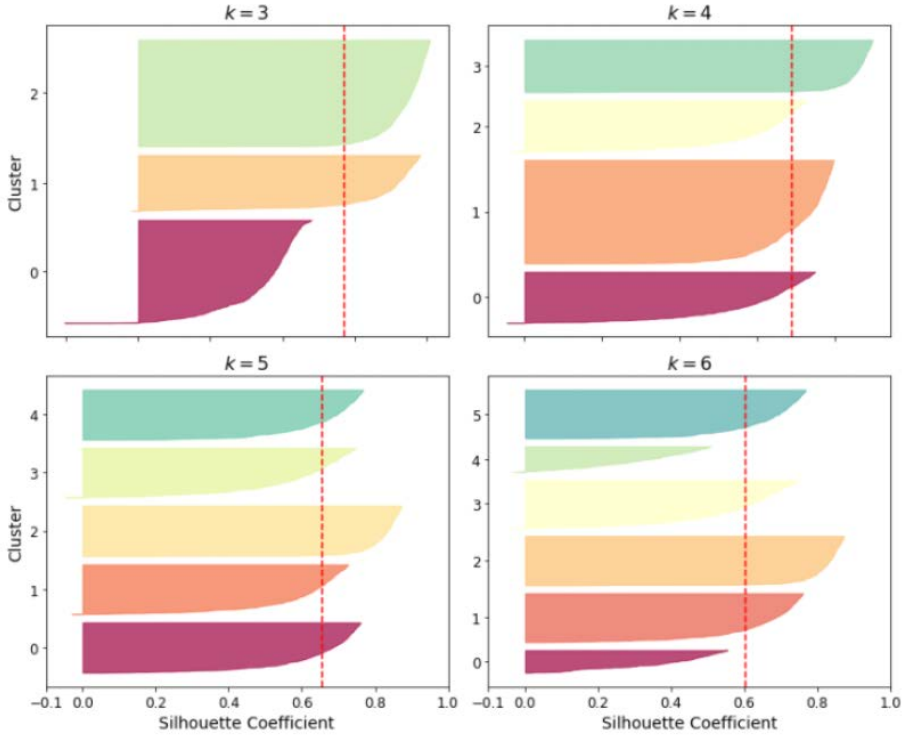
- K-means 알고리즘
  - 성능지표
    - 더 정확하지만 계산비용이 많이 드는 **실루엣 점수(Silhouette score)**를 사용
    - 각 군집간 거리가 얼마나 효율적으로 분리되어 있는지를 나타냄
      - 효율적 분리: 다른 군집과는 떨어져있고 동일 군집끼리의 데이터는 서로 가깝게 잘 뭉쳐있다는 것
    - $\frac{b(i)-a(i)}{max(a(i),b(i))}$ a: 같은 군집내에 있는 다른 샘플까지의 거리(즉, 내부 군집의 평균 거리), b: 해당 데이터가 속하지않은 가장 가까운 군집과 평균거리(가장 가까운 군집과의 평균거리)
    - [-1,1]사이의 값을 가지며 1에 가까울 수록 인접 군집에서 멀리 떨어져있고, 0에 가까울 수록 인접한 두 군집 사이의 결정 경계에 있다는 의미, 음 수 값은 잘못된 군집에 할당되었을 수 있음을 나타냄



실루엣 점수를 사용해 클러스터 개수 k를 선택하기

클러스터링

- K-means 알고리즘
  - 실루엣 다이어그램
    - 모든 샘플의 실루엣 계수를 할당된 군집과의 계수값으로 정렬하여 그린 그래프
    - 이 그래프의 높이는 군집이 포함하고 있는 샘플의 개수
    - 너비는 군집에 포함된 샘플의 정렬된 실루엣 계수를 나타냄
    - 수직 점선(파선)은 각 클러스터 개수에 해당하는 실루엣 점수



클러스터링

- DBSCAN
  - 밀도 기반 클러스터링 알고리즘
  - k-means처럼 단순히 거리만을 기준으로 군집화를 하는 것이 아니라 "가까이 있는 샘플들은 같은 군집에 속한다"는 원칙으로 군집을 차례로 넓혀가는 방식임
  - 샘플들의 몰려 있는 정도 즉, 밀도가 높은 부분을 중심으로 인접한 샘플들을 포함시켜 나감
  - 한 점을 기준점으로 반경 r내에 점이 n개 이상 있으면 하나의 군집으로 인식하는 방식임