머신 러닝 알고리즘

대표적 머신러닝 알고리즘

- k-최근접 이웃(k-Nearest Neighbor, kNN)
- 서포트 벡터머신(Support Vector Machine, SVM)
- 의사결정 트리(Decision Tree)
- 나이브 베이즈(Naïve Bayes)
- 앙상블(Ensemble)
- 군집화(Clustering)
- 주성분 분석(Principal Component Analysis, PCA)
- 선형 회귀(Linear Regression)
- 로지스틱 회귀(Logistic Regression)
- 딥러닝(Deep Neural Network, DNN)

대표적 머신러닝 알고리즘

- **k-최근접 이웃(k-Nearest Neighbor, kNN)**
- 서포트 벡터머신(Support Vector Machine, SVM)
- 의사결정 트리(Decision Tree)
- 나이브 베이즈(Naïve Bayes)
- 앙상블(Ensemble)
- 군집화(Clustering)
- 주성분 분석(Principal Component Analysis, PCA)
- 선형 회귀(Linear Regression)
- 로지스틱 회귀(Logistic Regression)
- 딥러닝(Deep Neural Network, DNN)

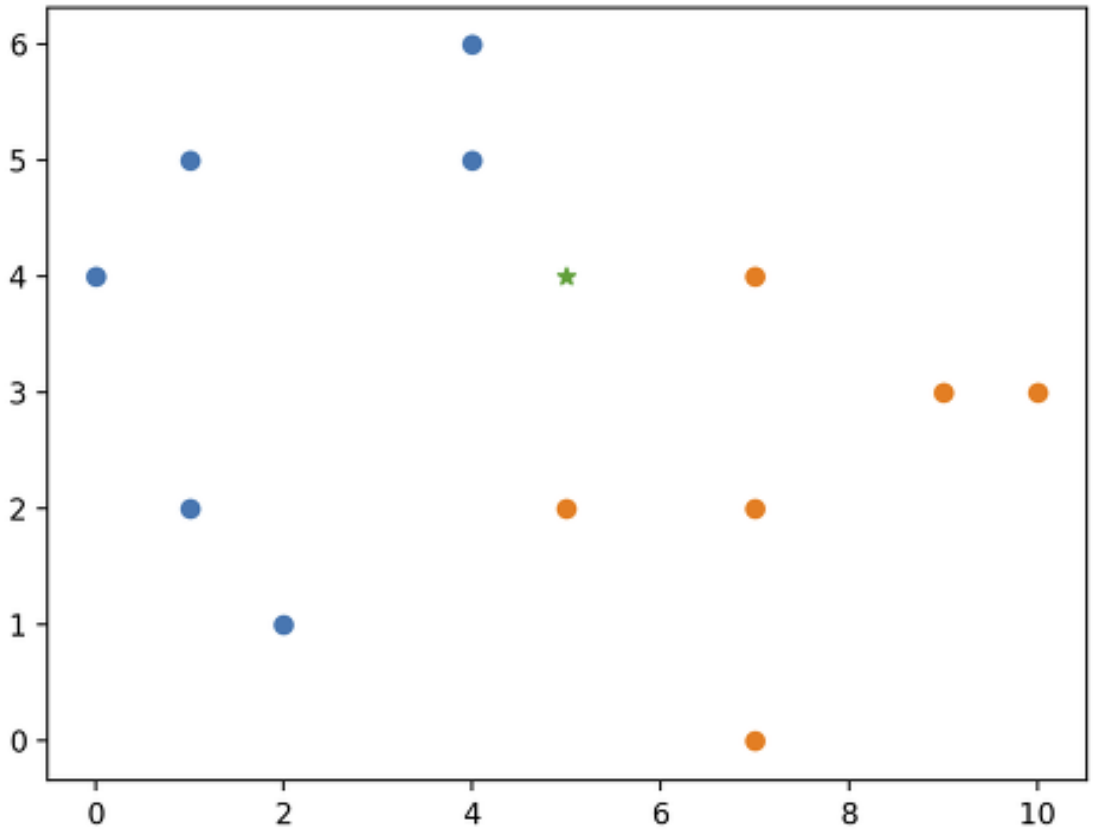
k-최근접 이웃(k-Nearest Neighbor, kNN)

- 특정 공간 내에서 입력과 제일 근접한 k개의 요소를 찾아, 더 많이 일치하는 것으로 분류하는 알고리즘.
- 데이터 분류에 사용되는 아주 간단한 지도학습 알고리즘.
  - 지도학습 : 머신러닝 학습 시 데이터와 함께 데이터에 대한 레이블(정답)을 함께 부여하는 학습 방식.
  - 데이터 분류 : 새로운 데이터를 기존 데이터의 레이블 중 하나로 분류하는 작업.

- 유사한 특성을 가진 데이터들끼리는 거리가 가깝다. 그리고 거리 공식을 사용하여 데이터 사이의 거리를 구할 수 있다.
- 분류기의 효과를 높이기 위해 파라미터를 조정할 수 있다.
- K-Nearest Neighbors의 경우 k 값을 변경할 수 있다.
- 분류기가 부적절하게 학습되면 overfitting 또는 underfitting이 나타날 수 있다.
- K-Nearest Neighbors의 경우 너무 작은 k는 overfitting, 너무 큰 k는 underfitting을 야기한다.

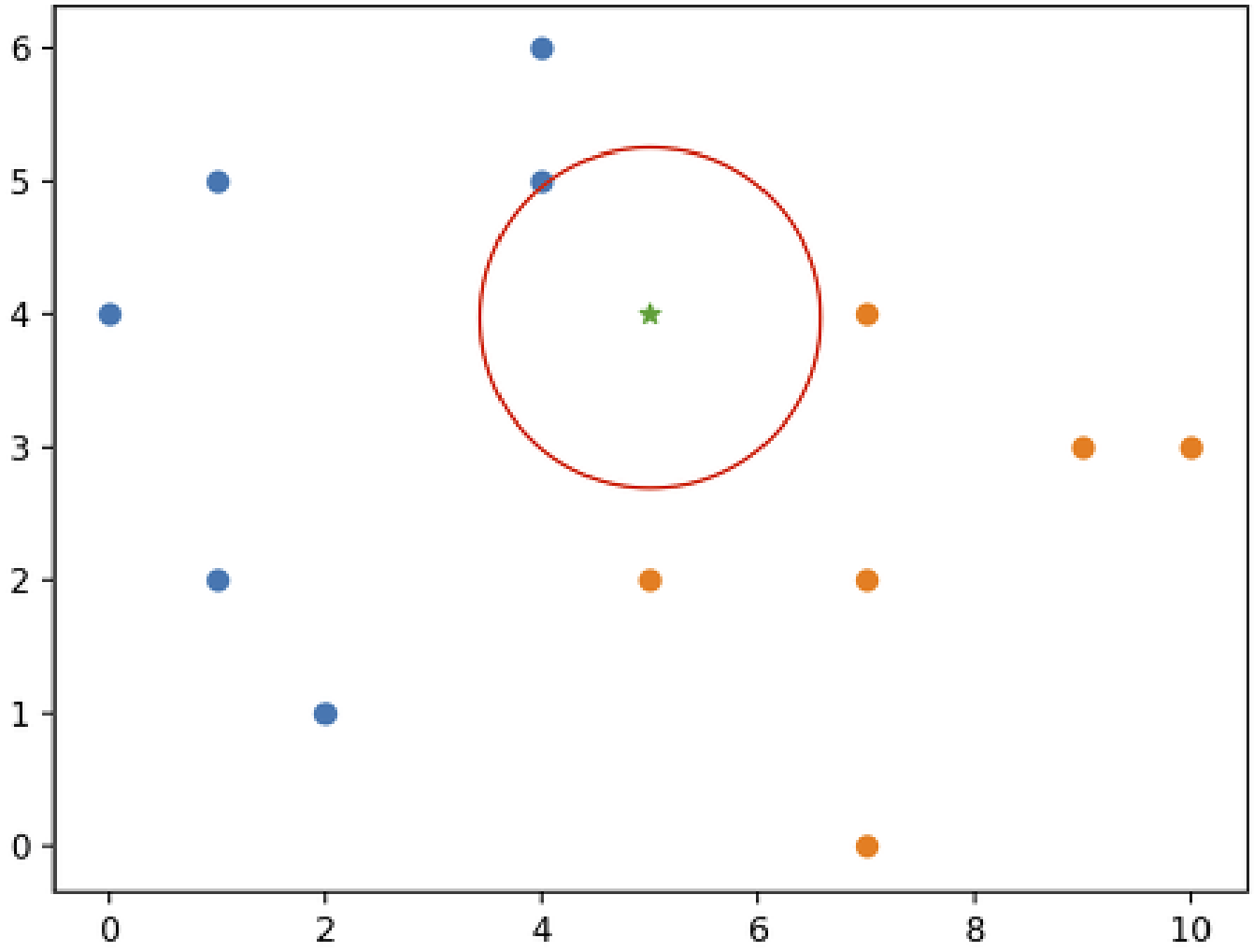
k-최근접 이웃(k-Nearest Neighbor, kNN)

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$



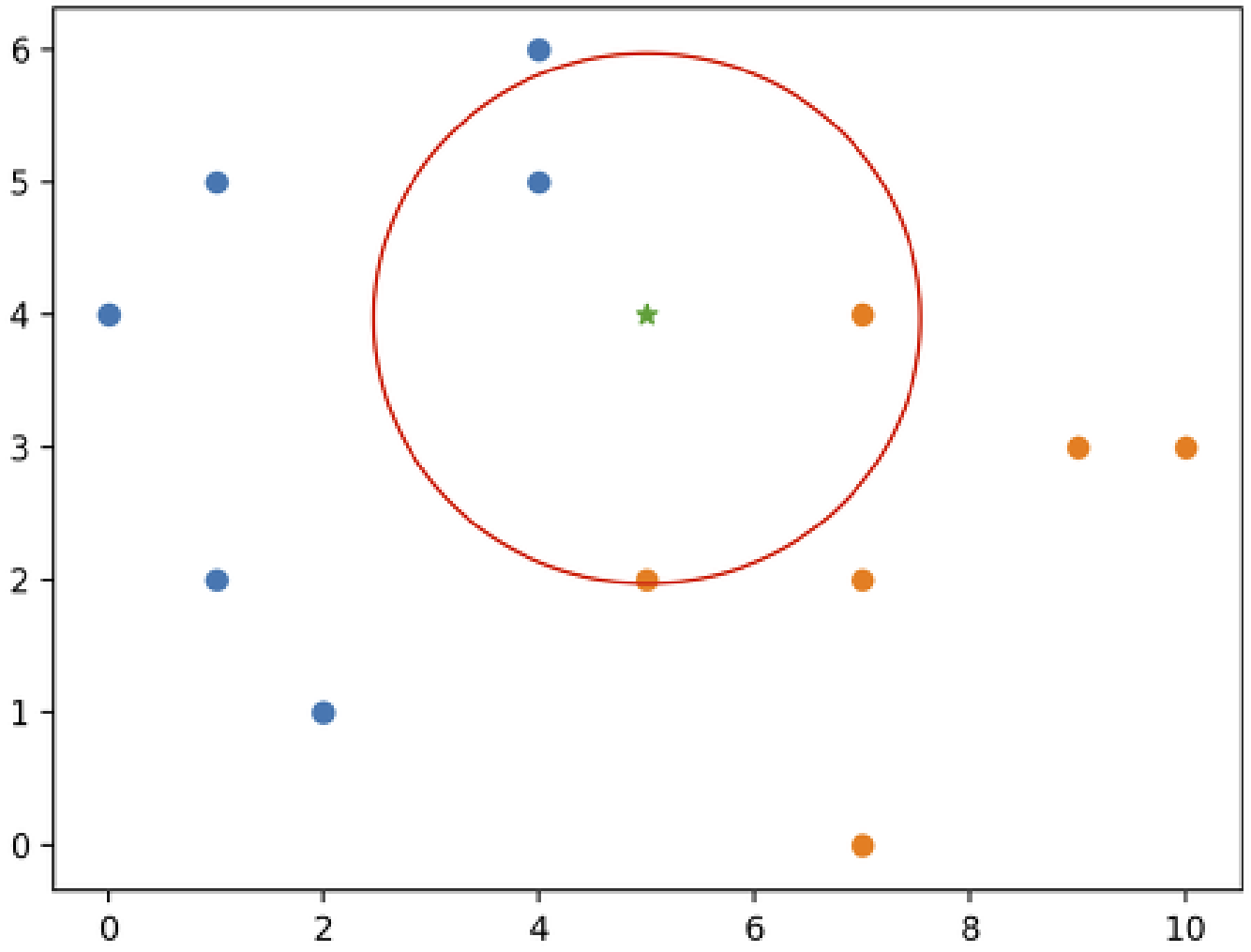
k-최근접 이웃(k-Nearest Neighbor, kNN)

- k = 1



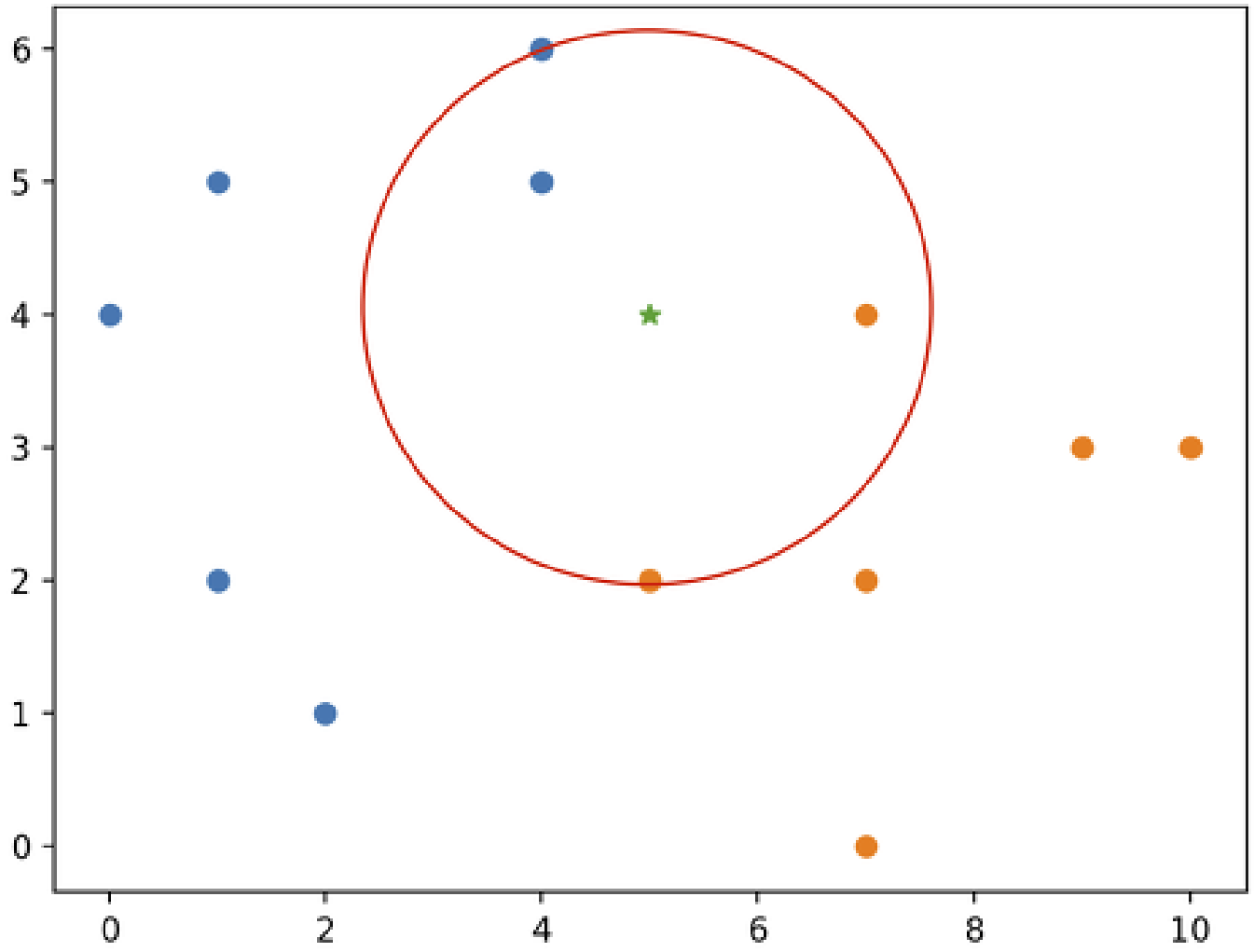
k-최근접 이웃(k-Nearest Neighbor, kNN)

- k = 3



k-최근접 이웃(k-Nearest Neighbor, kNN)

- k = 4



k-최근접 이웃(k-Nearest Neighbor, kNN)

- 장점
  - 구현이 쉽다.
  - 알고리즘을 이해하기 쉽다.
  - 별도의 모델 학습이 필요 없다.
  - 하이퍼 파라미터가 적다 – 이웃의 개수(k).

- 단점
  - 예측 속도가 느리다.
  - 메모리를 많이 쓴다.
  - 노이즈 데이터에 예민하다.

대표적 머신러닝 알고리즘

- k-최근접 이웃(k-Nearest Neighbor, kNN)
- **서포트 벡터머신(Support Vector Machine, SVM)**
- 의사결정 트리(Decision Tree)
- 나이브 베이즈(Naïve Bayes)
- 앙상블(Ensemble)
- 군집화(Clustering)
- 주성분 분석(Principal Component Analysis, PCA)
- 선형 회귀(Linear Regression)
- 로지스틱 회귀(Logistic Regression)
- 딥러닝(Deep Neural Network, DNN)

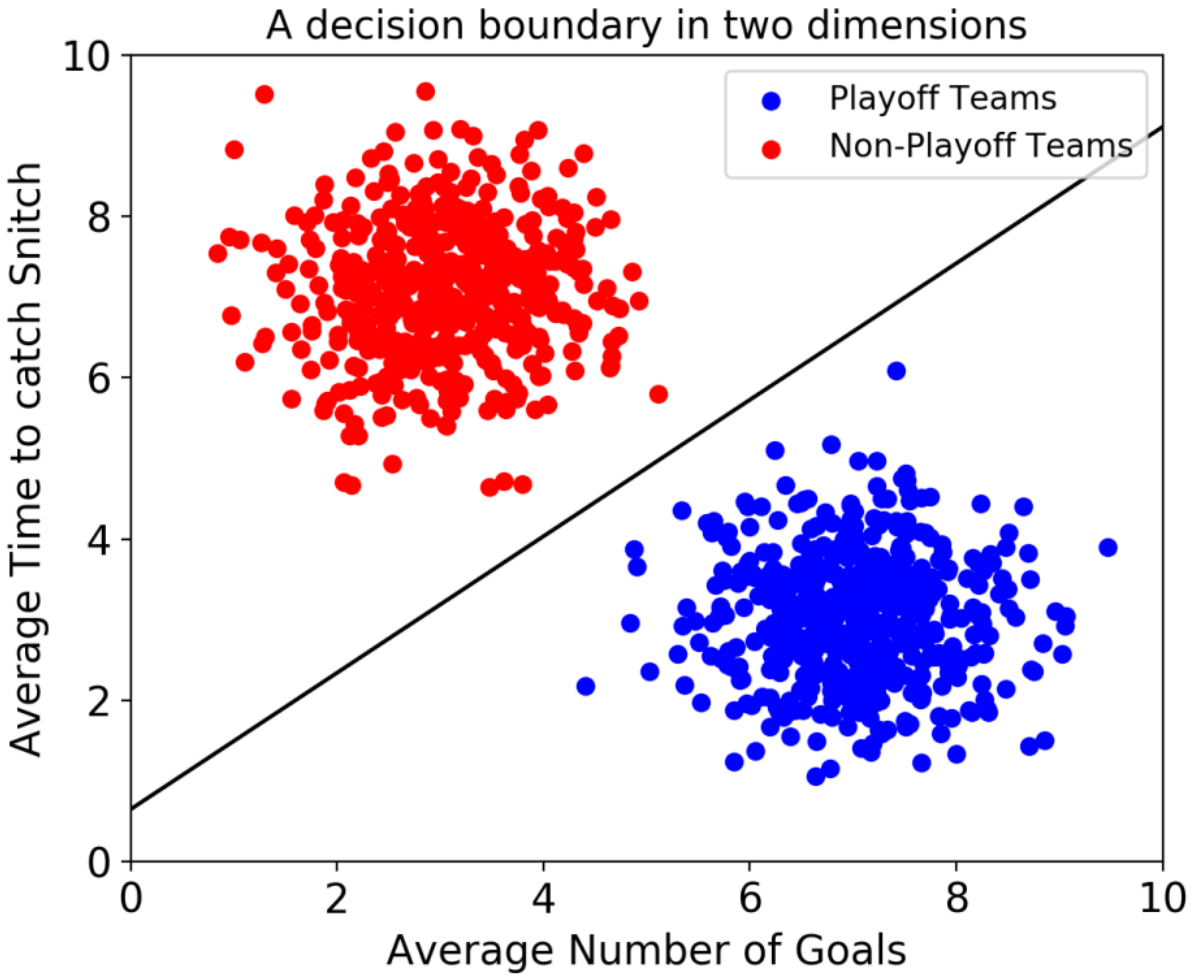
서포트 벡터머신(Support Vector Machine, SVM)

- 사용하기 편하면서도 높은 정확도를 보이는 데이터 분류를 위한 지도학습 머신러닝 알고리즘.
- 서포트 벡터를 사용해서 결정 경계를 정의하고, 분류되지 않은 점을 해당 결정 경계와 비교해서 분류.
- SVM의 중요 용어
  - **결정 경계(Decision Boundary)** : 서로 다른 분류 값을 결정하는 경계.
  - **서포트 벡터(support vector)** : 결정 경계선과 가장 가까이 맞닿은 데이터 포인트(클래스의 점 들).
  - **마진(margin) :** 서포트 벡터와 결정 경계 사이의 거리.

- **SVM의 목표는 바로 이 마진을 최대로 하는 결정 경계를 찾는 것.**

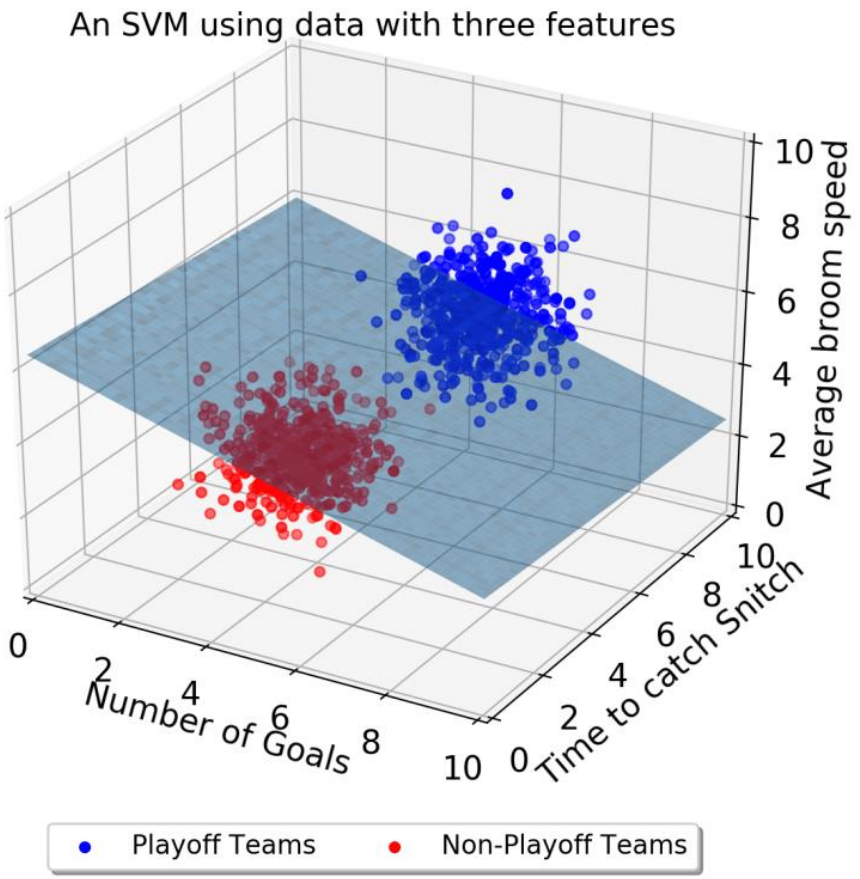
서포트 벡터머신 – 결정 경계(선)

- 데이터에 2개 속성(feature)만 있다면 **결정 경계**는 이렇게 간단한 선 형태



서포트 벡터머신 – 결정 경계(면)

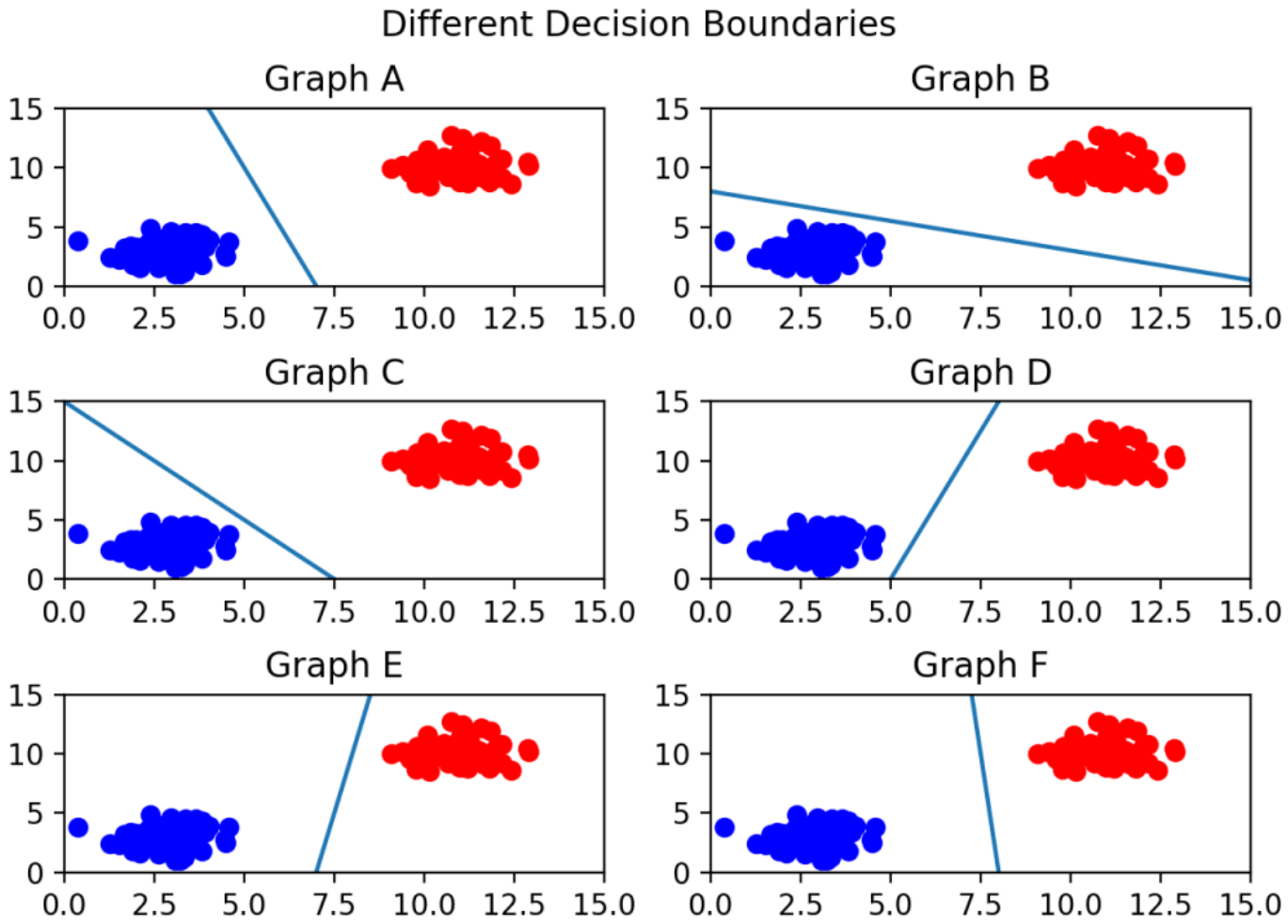
➢ 속성이 3개로 늘어난다면 이렇게 3차원으로 그려야 한다.



- 이때의 **결정 경계**는 '선'이 아닌 '평면'이 된다.
- 우리가 이렇게 시각적으로 인지할 수 있는 범위는 딱 3차원까지다.
- 차원, 즉 속성의 개수가 늘어날수록 당연히 복잡
- **결정 경계**도 단순한 평면이 아닌 고차원이 될 텐데 이를 **"초평면(hyperplane)"**이라고 부른다.

서포트 벡터머신 – 결정 경계

➢ **최적의 결정 경계(Decision Boundary)**



서포트 벡터머신 – 결정 경계

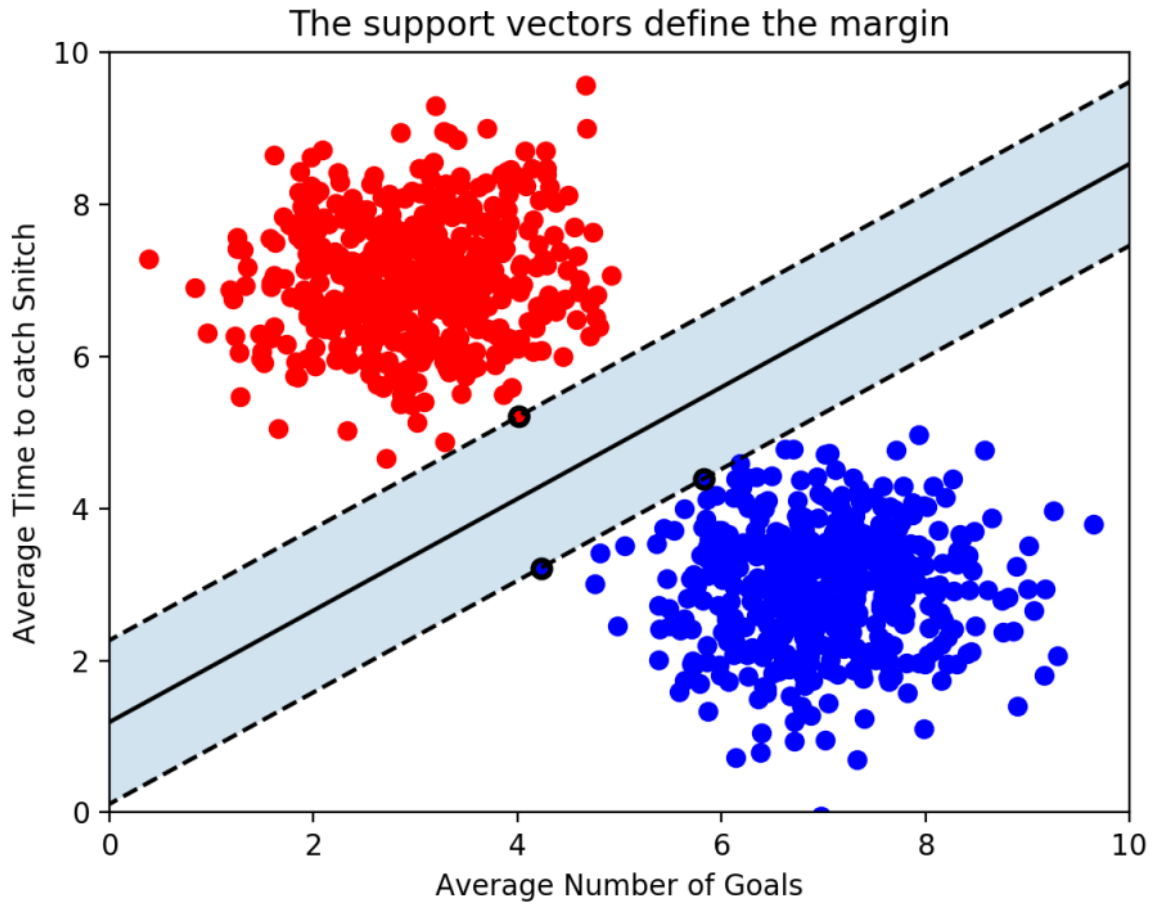
➢ **최적의 결정 경계(Decision Boundary)**

- 어떤 그래프가 제일 위태로워 보이는가?
    - C를 보면 선이 파란색 부류와 너무 가까워서 아슬아슬해 보인다.
- 그렇다면 어떤 결정 경계가 가장 적절해 보이는가?
    - 당연히 F다. 두 클래스(분류) 사이에서 거리가 가장 멀기 때문이다.
- **결정 경계는 데이터 군으로부터 최대한 멀리 떨어지는 게 좋다**는 걸 알았다.
- 실제로 서포트 벡터 머신(Support Vector Machine)이라는 이름에서 **Support Vectors는 결정 경계와 가까이 있는 데이터 포인트들을 의미한다.**
- 이 데이터들이 경계를 정의하는 결정적인 역할을 하는 셈이다.

서포트 벡터머신 - 마진

➢ 마진(Margin)

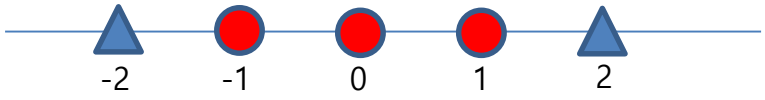
- **결정 경계와 서포트 벡터 사이의 거리**를 의미한다.



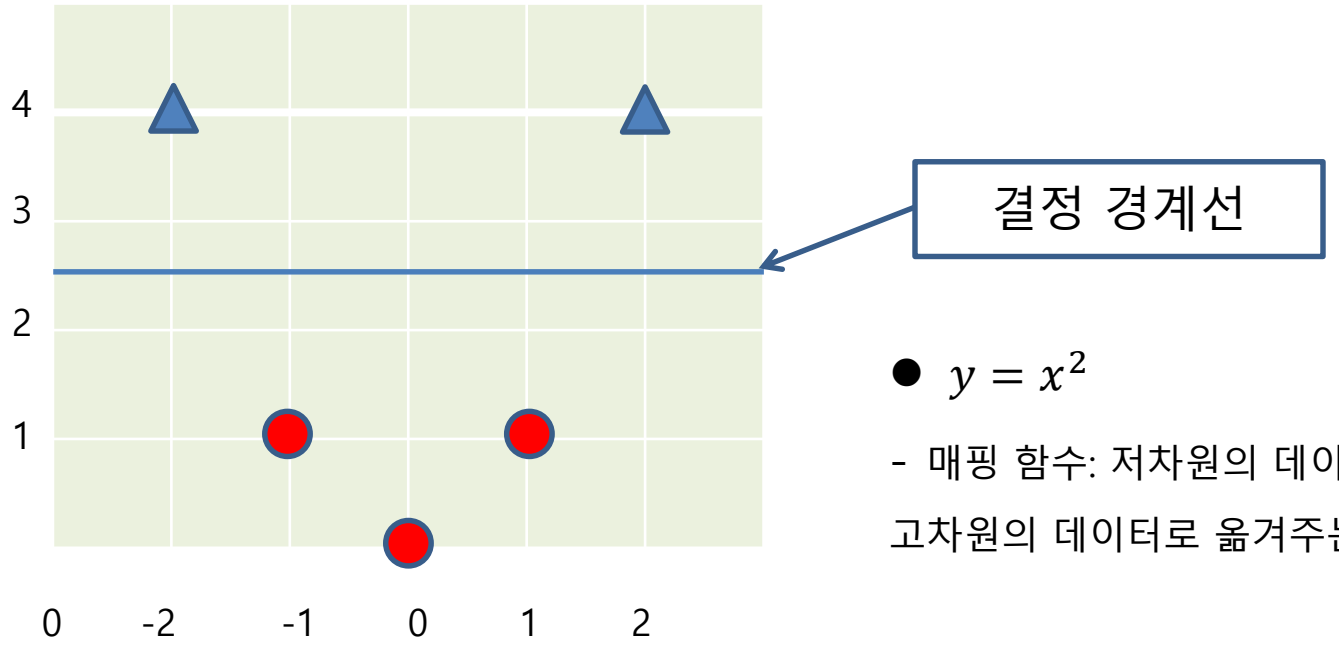
- 가운데 실선이 하나 그어져있는데, 이게 바로 '결정 경계'
- 그리고 그 실선으로부터 검은 테두리가 있는 빨간점 1개, 파란점 2개까지 영역을 두고 점선을 그어놓았다. 이게 바로 '마진(margin)'
- **최적의 결정 경계는 마진을 최대화한다.**
- 옆 그림에서는 x축과 y축 2개의 속성을 가진 데이터로 결정 경계를 그었는데, 총 3개의 데이터 포인트(서포트 벡터)가 필요했다. 즉, **n개의 속성을 가진 데이터에는 최소 n+1개의 서포트 벡터가 존재한다**는 걸 알 수 있다.

서포트 벡터머신 – 커널 트릭

- 1차원의 데이터 결정 경계 찾기



- 2차원 공간으로 옮겨진 데이터 결정 경계 찾기

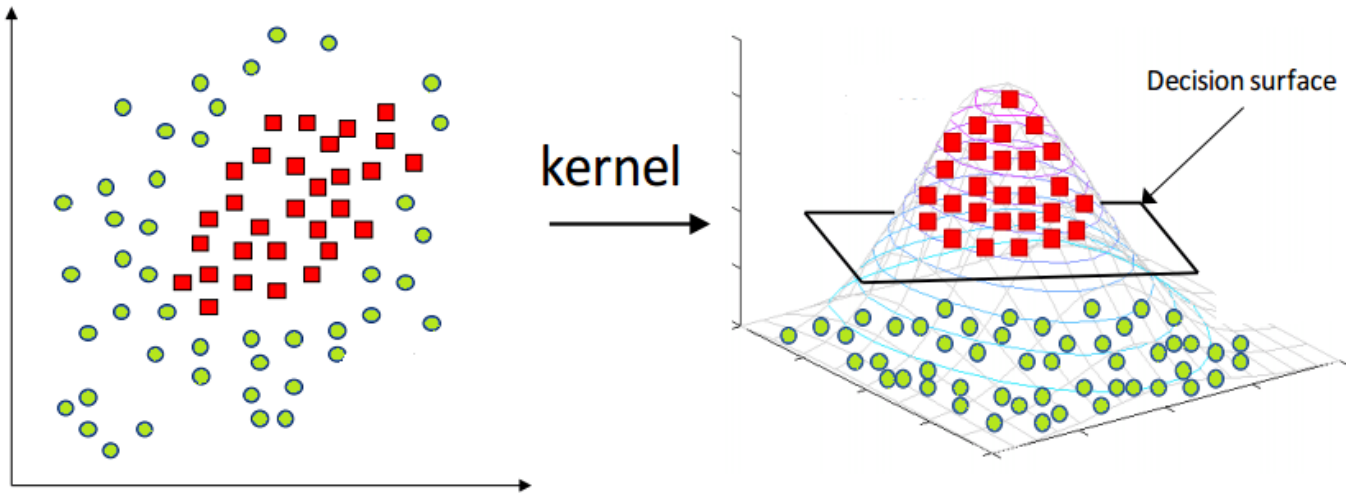


- $y = x^2$

- 매핑 함수: 저차원의 데이터를 고차원의 데이터로 옮겨주는 함수.

서포트 벡터머신 – 커널 트릭

- 커널 트릭(kernel trick)
  - 매핑 함수를 가지고 실제로 많은 양의 데이터를 저차원에서 고차원으로 옮기기에는 계산량이 너무 많아서 현실적으로 사용하기가 어려움.
  - 데이터를 고차원으로 보내진 않지만 보낸 것과 동일한 효과를 줘서 매우 빠른 속도로 결정 경계선을 찾는 방법.
  - 저차원에서 결정 경계를 찾지 못할 때 고차원으로 데이터(벡터)를 옮겨서 결정 경계를 찾는 방법.



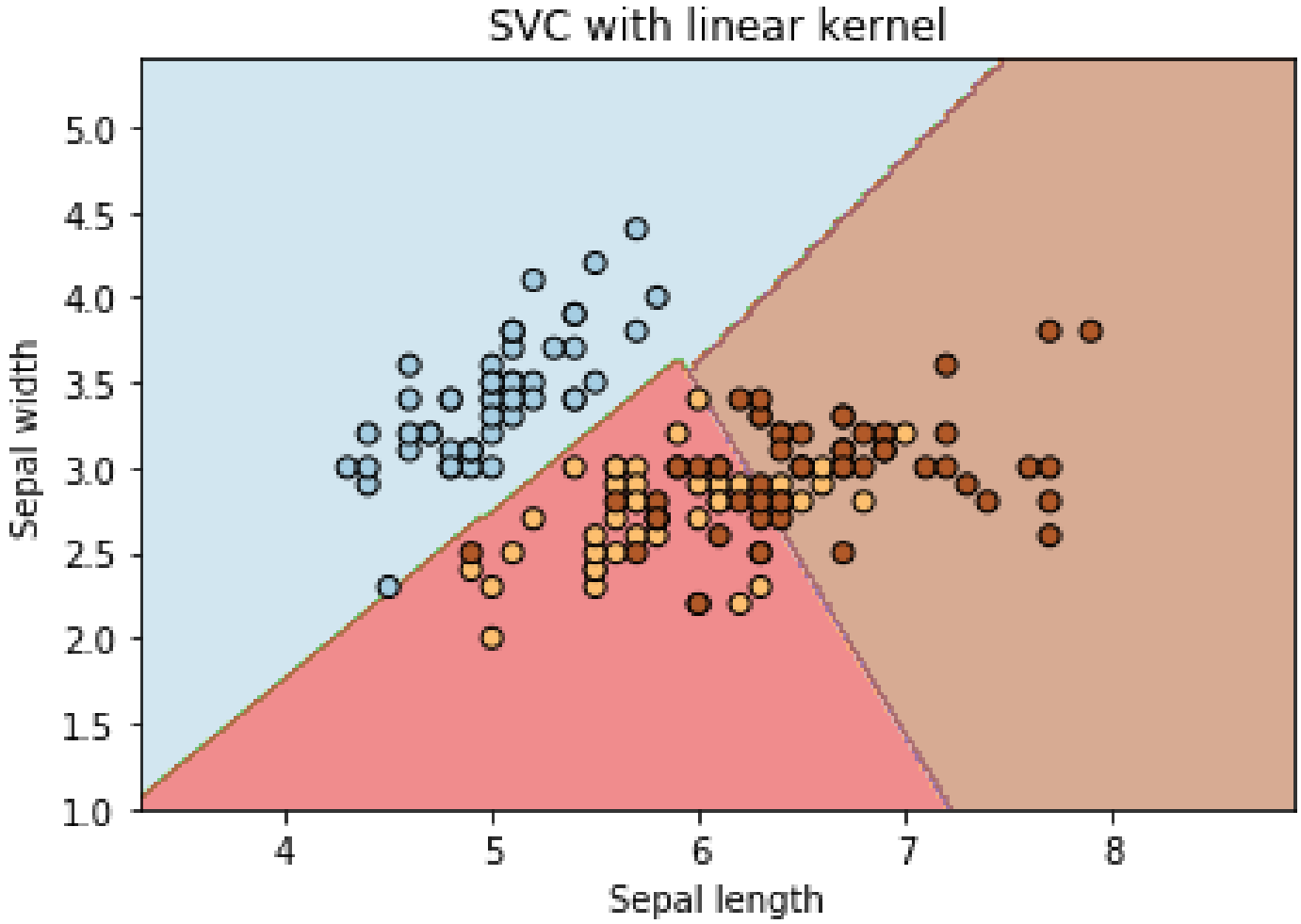
서포트 벡터머신 – 파라미터 튜닝

- 결정 경계 = N – 1 차원 (단, 데이터의 벡터 공간은 N차원)
- 따라서, 2차원 공간의 결정 경계는 선, 3차원 공간의 결정 경계는 면.
- 파라미터 **C**(cost, 비용)는 허용되는 오류 양을 조절.
  - 마진의 너비를 조정하는 파라미터.
  - C 값이 클수록, 마진은 낮아지고, 학습 에러율은 감소하는 방향으로 결정 경계선을 만들고, 이를 **하드 마진(hard margin)**이라 부름 – 너무 높으면 과대 적합의 위험.
  - C 값이 작을수록, 마진을 최대한 높이고, 학습 에러율을 증가시키는 방향으로 결정 경계선을 만들어서 **소프트 마진(soft margin)**을 만듦 – 너무 낮으면 과소 적합의 위험.

서포트 벡터머신 – 파라미터 튜닝

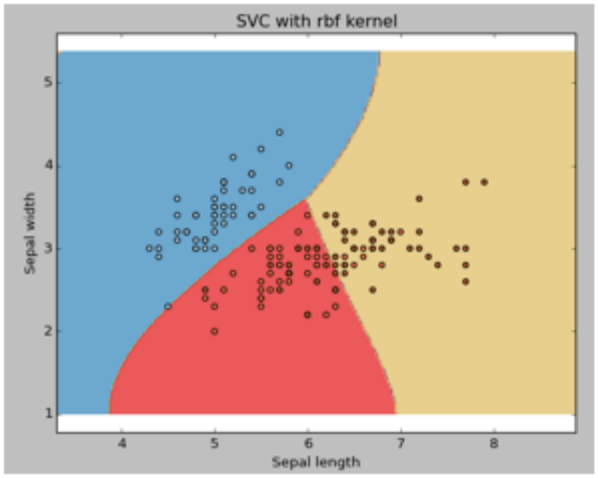
- SVM에서는 선형으로 분리할 수 없는 점들을 분류하기 위해 **커널(kernel)**을 사용.
- 커널(kernel)은 원래 가지고 있는 데이터를 더 높은 차원의 데이터로 변환.
  - 2차원의 점으로 나타낼 수 있는 데이터를
    - 다항식(polynomial) 커널은 3차원으로,
    - RBF 커널은 점을 무한한 차원으로 변환.
- RBF 커널에는 파라미터 감마(gamma)가 있음.
  - 커널의 데이터포인트 표준 편차를 결정하는 조절 변수.
  - 감마가 너무 크면 학습 데이터에 너무 의존해서 오버피팅이 발생할 수 있다.
- 대부분의 머신러닝 지도 학습 알고리즘은 학습 데이터 모두를 사용하여 모델을 학습.
  - SVM에서는 결정 경계를 정의하는 게 결국 서포트 벡터이기 때문에, 데이터 포인트 중에서 서포트 벡터만 잘 골라내면 나머지 쓸 데 없는 수많은 데이터 포인트들을 무시할 수 있음.
  - 그래서 매우 빠르다.

서포트 벡터머신 – 파라미터 튜닝

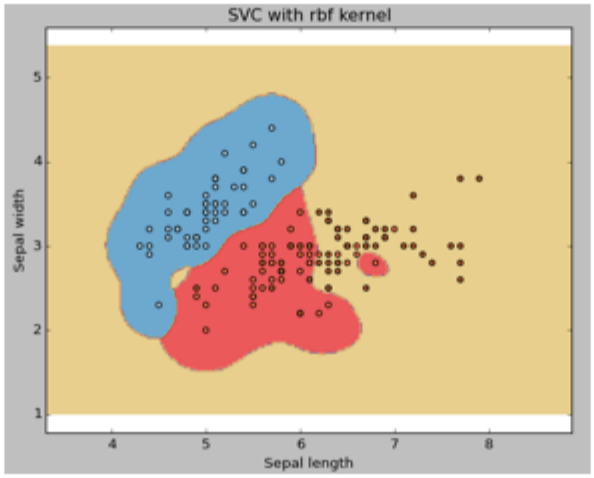


서포트 벡터머신 – 파라미터 튜닝

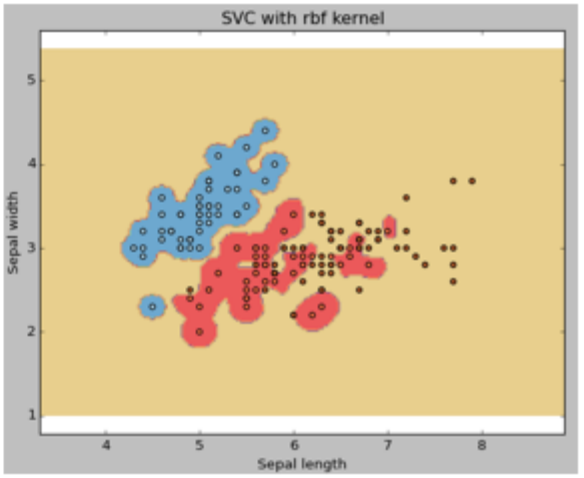
gamma =0



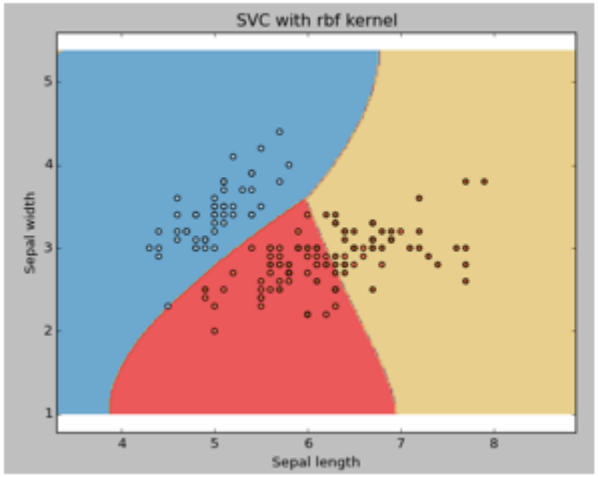
gamma =10



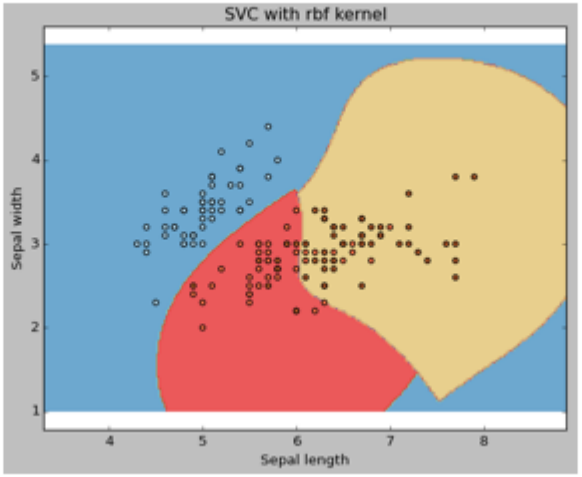
gamma =100



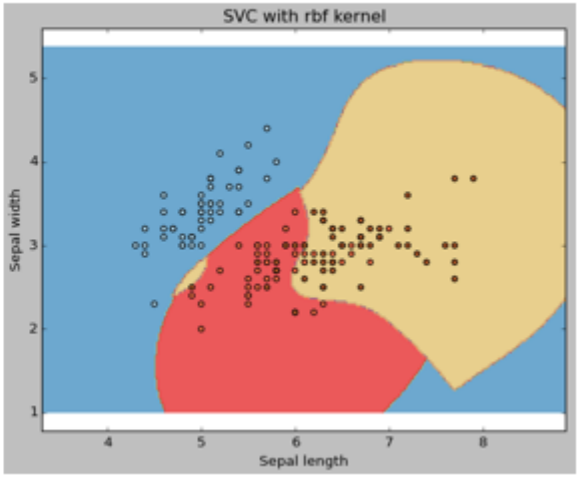
c =1



C =100



c =1000



서포트 벡터머신 – 장/단점

➢ 장점

- 커널 트릭을 사용함으로써 특성이 다양한 데이터를 분류하는 데 강하다.
  - N개의 특성을 가진 데이터는 N차원 공간의 데이터 포인트로 표현되고, N차원 공간 또는 그 이상의 공간에서 초평면을 찾아 데이터 분류가 가능하기 때문.
- 예측 속도가 빠르다.
- 파라미터(C, gamma)를 조정해서 과대적합 및 과소적합에 대처할 수 있다.
- 적은 학습 데이터로도 딥러닝만큼 정확도가 높은 분류를 기대할 수 있다.

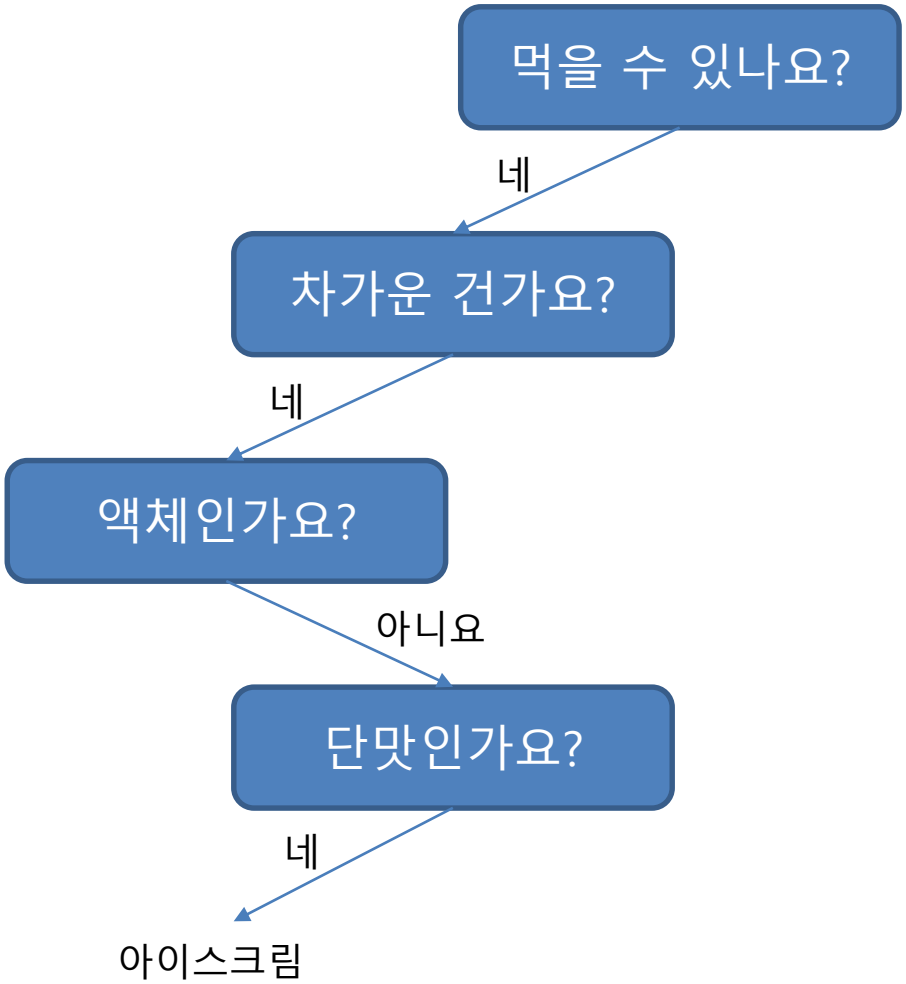
➢ 단점

- 데이터 전처리 과정(data preprocessing)이 상당히 중요하다.
  - 특성이 비슷한 수치로 구성된 데이터 같은 경우에는 SVM을 쉽게 활용할 수 있지만, 특성이 다양하거나 혹은 확연히 다른 경우에는 데이터 전처리 과정을 통해 데이터 특성 그대로 벡터 공간에 표현해야 함.
- 특성이 많을 경우 결정 경계 및 데이터의 시각화가 어려움 – SVM의 분류 결과를 이해하기 힘듦.
- 커널 트릭 오사용시 과대적합되기 쉽다.

대표적 머신러닝 알고리즘

- k-최근접 이웃(k-Nearest Neighbor, kNN)
- 서포트 벡터머신(Support Vector Machine, SVM)
- **의사결정 트리(Decision Tree)**
- 나이브 베이즈(Naïve Bayes)
- 앙상블(Ensemble)
- 군집화(Clustering)
- 주성분 분석(Principal Component Analysis, PCA)
- 선형 회귀(Linear Regression)
- 로지스틱 회귀(Logistic Regression)
- 딥러닝(Deep Neural Network, DNN)

의사결정 트리(Decision Tree)



의사결정 트리(Decision Tree)

- 데이터 분류 및 회귀에 사용되는 지도학습 알고리즘.
- **데이터의 특징 속에서 분류에 큰 영향을 끼치는 특징을 발견하고, 상위 노드로 선택하는 알고리즘.**
- 머신러닝에서 의미 있는 질문이란 데이터의 특징 중 의미 있는 특징에 해당하며, 의사결정 트리의 핵심은 바로 영향력이 큰 특징을 상위 노드로, 영향력이 작은 특징은 하위 노드로 선택하는 것.
- 데이터가 주어졌을 때 의사결정 트리는 어떻게 특징별 영향력의 크고 작음을 비교할 수 있을까?
  - 크고 작음을 비교하기위해 수치적인 결과가 필요.

의사결정 트리 알고리즘과 정보 엔트로피의 관계

- 스무고개를 할 때 질문자는 질문을 던질 때마다 약간씩의 정보를 획득.
- 약간씩의 정보를 획득하는 과정은 정답에 대한 불확실성이 조금씩 줄어든다는 것과 같은 개념.
- **엔트로피(Entropy)** : 정보 이론(information Theory)에서는 이 불확실성을 수치적으로 표현한 값.
- **정보 이득(information gain)** : 불확실성이 줄어 든 정도. 즉 질문 이전의 엔트로피에서 질문 후의 엔트로피를 뺀 값.
  - $Gain(T, X) = Entropy(T) - Entropy(T, X)$
- 확률을 바탕으로 정보 엔트로피를 구하는 공식
  - $Entropy = \sum_{i=1}^{n} -p_i \, log_2 p_i$

의사결정 트리 사례로 정보 엔트로피 구하기

이름	군필	긴생머리	성별
김철수	네	아니오	남자
이영희	아니오	아니오	여자
홍길동	네	아니오	남자
최빛나	아니오	네	여자
강감찬	네	아니오	남자
유관순	아니요	아니오	여자

- 엔트로피 = -p(남자) * log(p(남자)) – p(여자) * log(p(여자))
- 의사결정 트리의 최초 엔트로피는 1이며, 계산 방법은
  - – (3/6) * log(3/6) – (3/6) * log(3/6) = 1

한 가지 특징에 대한 엔트로피 계산

$$Entropy = \sum_{c \in X} P(c)E(c)$$

- `X` : 선택된 특징.
- `c` : 선택된 특징에 의해 생성된 하위 노드.
- `P(c)`: 선택된 특징에 의해 생성된 하위 노드에 데이터가 속할 확률.
- `E(c)`: 선택된 특징에 의해 생성된 하위 노드의 엔트로피.

- 군대라는 특징으로 데이터를 분리했을 때의 엔트로피

  3/6 * E[3,0] + 3/6 * E[0,3]
  = 3/6 * (-(3/3) * log(3/3) – (0/3) * log(0/3)) + 3/6 * (-(0/3) * log(0/3) – (3/3) * log(3/3))
  = 0

- 긴 생머리라는 특징으로 데이터를 분리했을 때의 엔트로피

  1/6 * E[0,1] + 5/6 * E[3,2]
  = 1/6 * (-(0/1) * log(0/1) – (1/1) * log(1/1)) + 5/6 * (-(3/5) * log(3/5) – (2/5) * log(2/5))
  = 0.966

- "군대"로 데이터를 분리할 때의 정보 이득 : 1
- "긴 생머리"로 데이터를 분리할 때의 정보 이득 : 0.034

지니 계수(Gini coefficient)

- 특징에 의한 분리가 이진 분류로 나타날 경우 지니 계수를 사용할 수 있음.
- 사이킷런의 의사결정 트리는 CART(classification and regression tree) 타입의 의사결정 트리.
- CART는 트리의 노드마다 특징을 이진 분류하는 특징이 있기에 사이킷런은 트리를 구성할 때 기본적으로 지니 계수를 사용.
- 지니 계수의 특징
  - 특징이 항상 이진 분류로 나뉠 때 사용됨.
  - 지니 계수가 높을수록 순도가 높음.
    - 순도가 높다는 뜻: 한 그룹에 모여있는 데이터들의 속성들이 많이 일치한다는 뜻.
    - 불순도가 높다는 뜻: 한 그룹에 여러 속성의 데이터가 많이 섞여 있다는 뜻.
- 의사결정 트리 알고리즘은 지니 계수가 높은 특징으로 의사결정 트리 노드를 결정.

지니 계수를 통한 의사결정 트리 노드 결정 순서

1. 특징으로 분리된 두 노드의 지니 계수를 구함(P^2 + Q^2)
2. 특징에 대한 지니 계수를 구함.

예) 군대라는 특징으로 "군대를 다녀옴", "다녀오지 않음"으로 구분할 경우 지니 계수

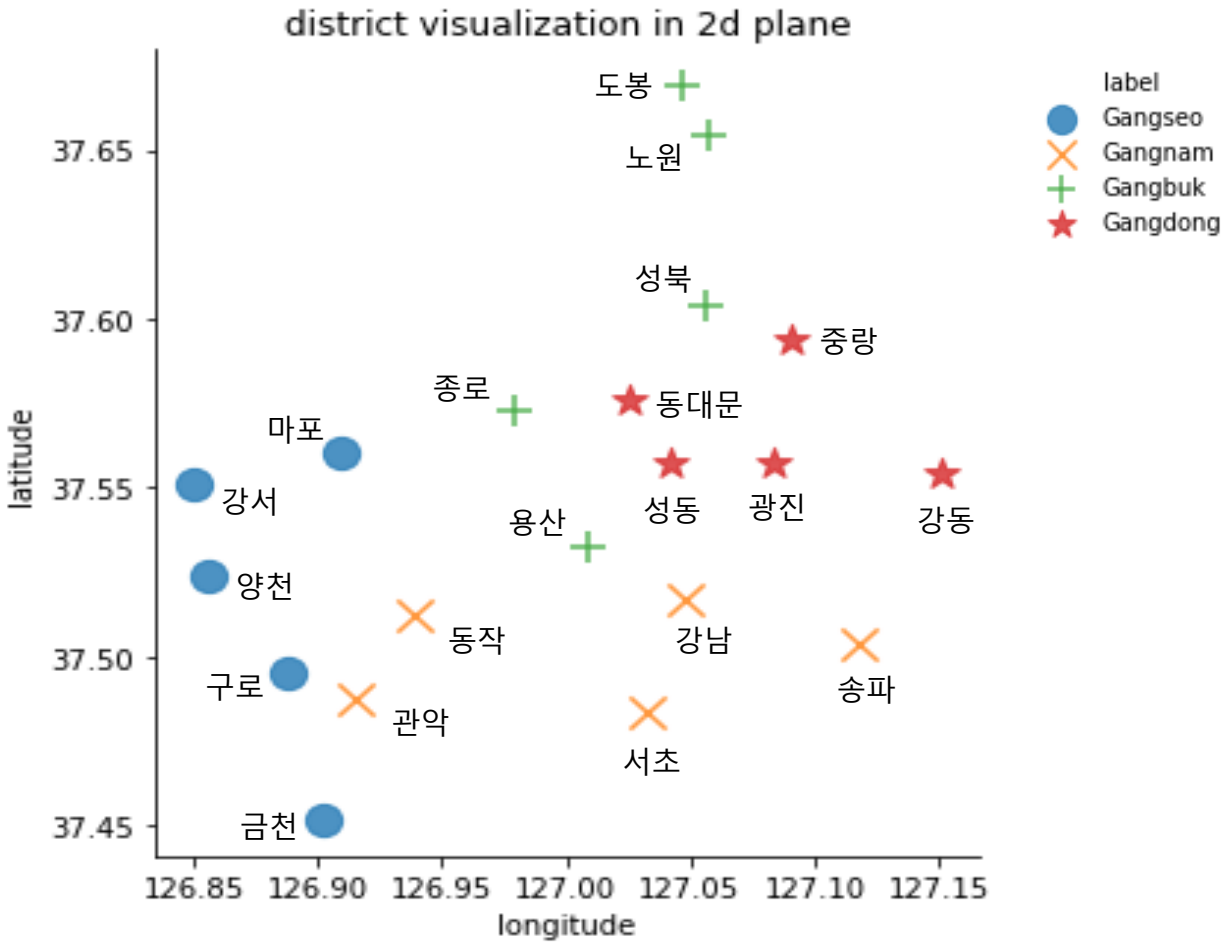
- 군대를 다녀옴 : (0/3)^2 + (3/3)^2 = 1
- 군대를 다녀오지 않음 : (3/3)^2 + (0/3)^2 = 1
- 군대 특징의 지니 계수 : 3/6 * 1 + 3/6 * 1 = 1

예) 긴 생머리를 특징으로 "긴 생머리", "긴 생머리 아님"으로 구분할 경우 지니 계수

- 긴생머리 : (0/1)^2 + (1/1)^2 = 1
- 긴생머리 아님 : (3/5)^2 + (2/5)^2 = 13/25
- 긴생머리 특징의 지니 계수 : 1/6 * 1 + 5/6 * 13/25 = 0.6

다중 분류

- 서울시 행정구역에 대한 다중 분류 데이터 시각화
  - 데이터를, 특징을 바탕으로 한 공간에 데이터 특징을 시각화함으로써, 우리는 머신러닝 학습에 필요한 특징과 불필요한 특징을 쉽게 구분 지을 수 있고, 데이터의 패턴을 눈으로 쉽게 파악할 수 있음.



다중 분류

➢ 문제 정의

- 서울 지역(구)의 경도와 위도 정보를 사용하여, 임의로 입력된 지역(동)을 강동, 강서, 강남, 강북으로 분류해보는 예제.

➢ 데이터 수집

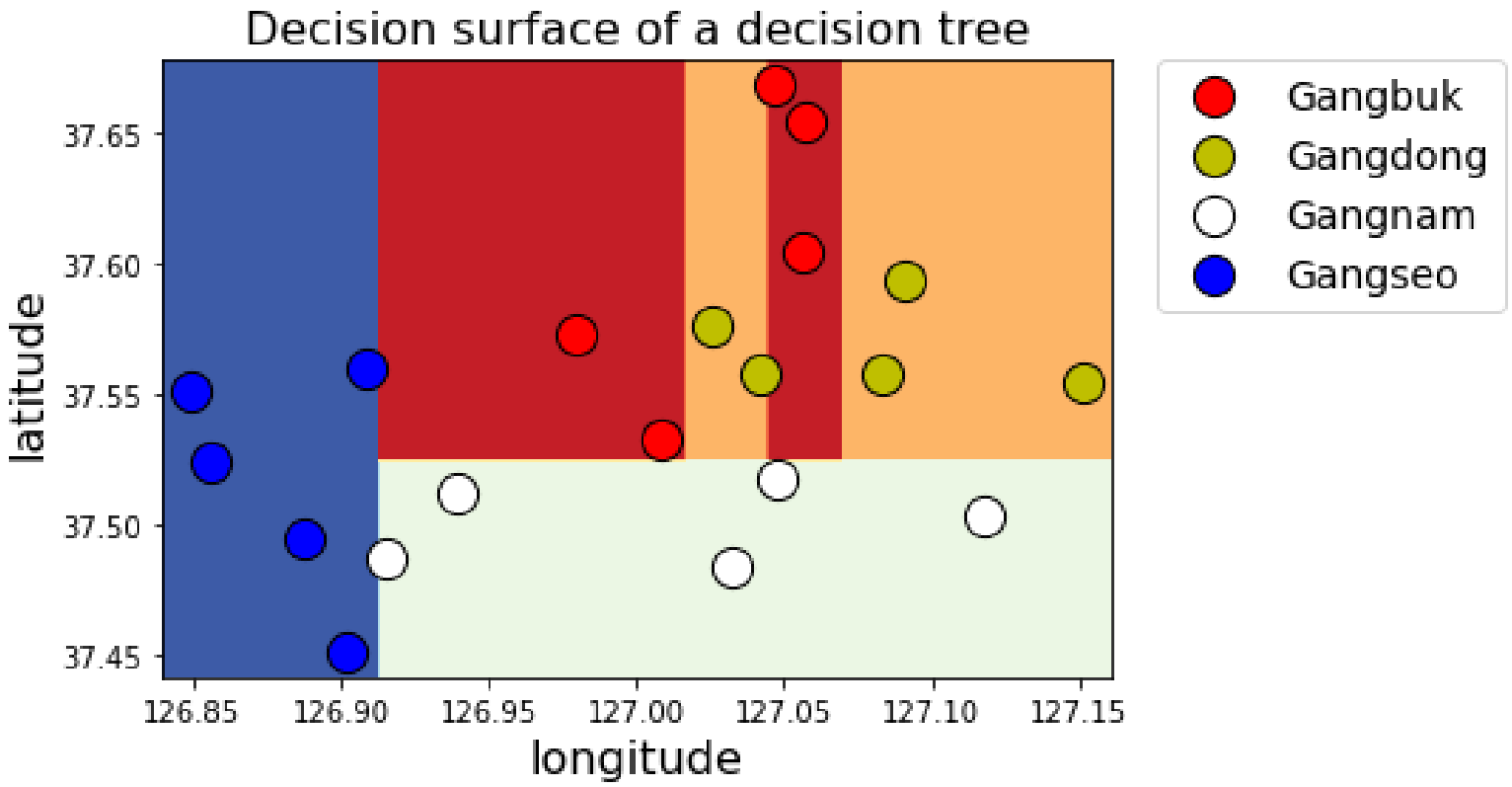
- 서울의 대표적인 구(district) 위치 데이터.
- 구(district) 정보는 학습에 사용

➢ 컬럼 주석

- **district**: 행정구역 (서초구, 송파구, 용산구 등, 서울의 단위 지역 분류)
- **dong**: 구(district)보다 작은 행정구역 (대치동, 도곡동, 암사동 등, 서울의 소단위 분류)
- **longitude**: 경도
- **latitude**: 위도
- **label**: 한강 기준으로 동,서,남,북으로 구분한 지역 명칭(강동,강서,강남,강북)

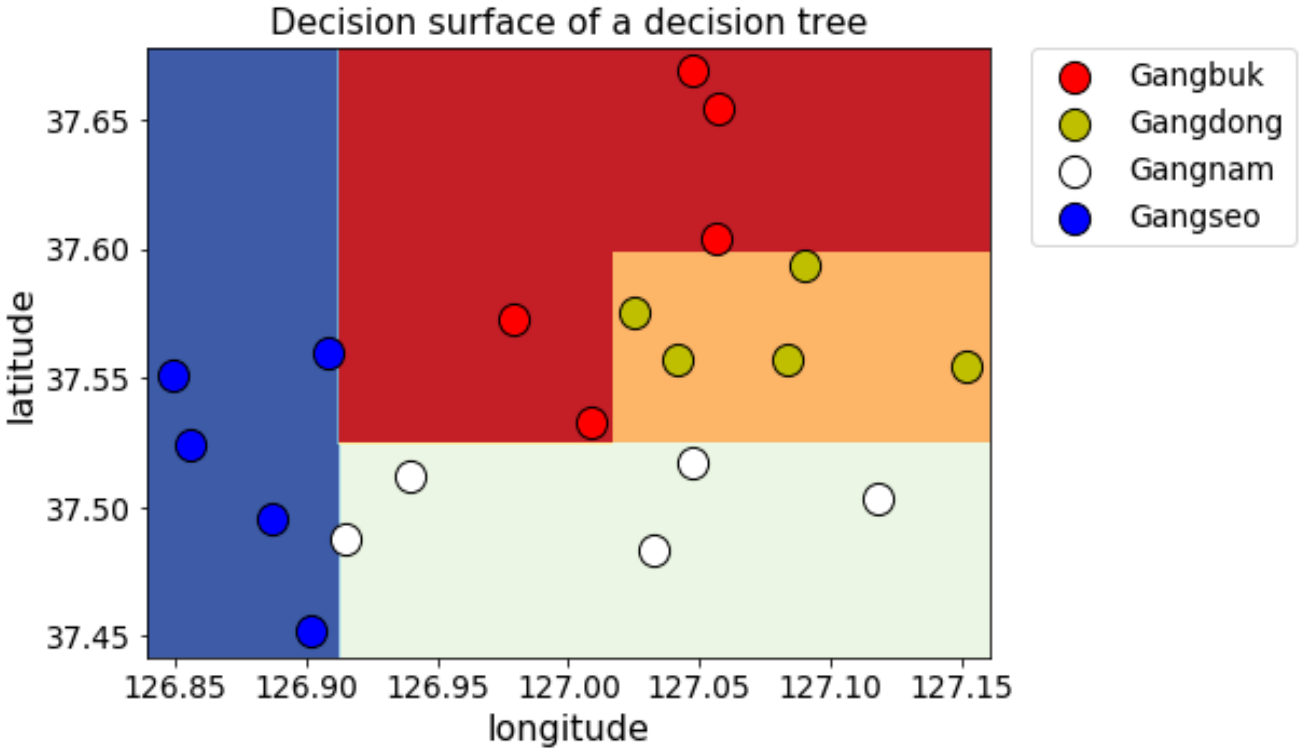
파라미터 없이 학습한 모델의 결정 표면 시각화

- 사이킷런의 의사결정 트리 알고리즘 적용 시, 특별히 파라미터를 설정하지 않고 학습할 경우 모델은 과대 적합 될 가능성이 높다.



파라미터 설정한 모델의 결정 표면 시각화

➢ **사이킷런의 의사결정 트리 모델**

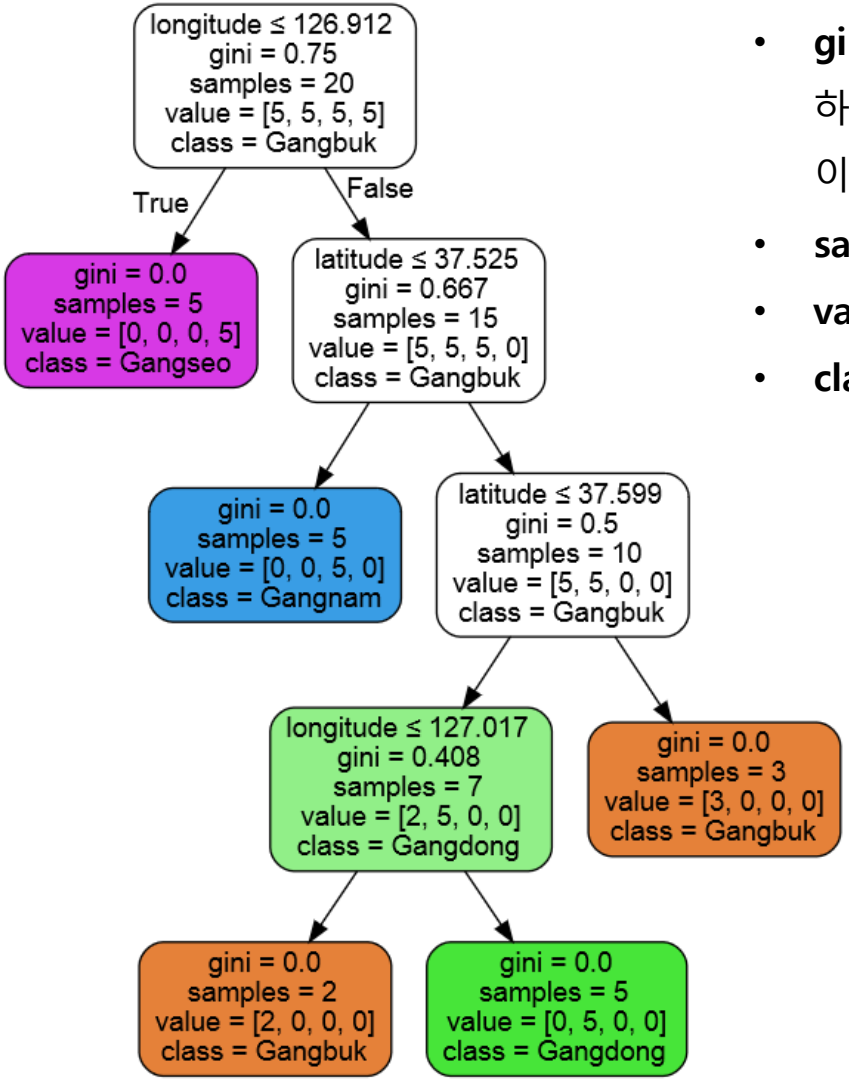



```
clf = tree.DecisionTreeClassifier(
                max_depth=4,
                min_samples_split=2,
                min_samples_leaf=2,
                random_state=70).fit(X_train, y_encoded.ravel())
```

- **max_depth** : 트리의 최대 한도 깊이.
- **min_samples_split** : 자식 노드를 갖기 위한 최소한의 데이터 개수.
- **mins_samples_leaf** : 맨 마지막 끝 노드의 최소 데이터 개수.
- **random_state** : 여러 번 실행해도, 파라미터가 같을 경우, 결과가 항상 같게 만들어주는 파라미터.

의사결정 트리 시각화

➢ 의사결정트리 시각화



- **gini** : 불순도 척도. 0일 경우 모든 샘플이 하나의 하나의 레이블을 가지며, 1에 가까울수록 여러 레이블이 한 노드에 존재함.
- **samples** : 노드 안에 들어 있는 데이터의 개수.
- **value** : 레이블별 데이터의 개수.
- **class** : 레이블.

의사결정 트리 – 장/단점

- 장점
  - 수학적인 지식이 없어도 결과를 해석하고 이해하기 쉽다.
  - 수치 데이터 및 범주 데이터에 모두 사용 가능하다.
  - 모델의 추론 과정을 시각화하기 쉽다.
  - 데이터에서 중요한 특성이 무엇인지 쉽게 알아낼 수 있다.

- 단점
  - 과대적합의 위험이 높다.
    - 의사결정 트리 학습 시 적절한 리프 노드의 샘플 개수와 트리의 깊이에 제한을 둬서 학습 데이터에 너무 모델이 치우치지 않게 주의.
  - 조정해야 할 하이퍼 파라미터가 많다.

대표적 머신러닝 알고리즘

- k-최근접 이웃(k-Nearest Neighbor, kNN)
- 서포트 벡터머신(Support Vector Machine, SVM)
- 의사결정 트리(Decision Tree)
- **나이브 베이즈(Naïve Bayes)**
- 앙상블(Ensemble)
- 군집화(Clustering)
- 주성분 분석(Principal Component Analysis, PCA)
- 선형 회귀(Linear Regression)
- 로지스틱 회귀(Logistic Regression)
- 딥러닝(Deep Neural Network, DNN)

나이브 베이즈(Naïve Bayes)

- 확률 기반 머신러닝 분류 알고리즘.
- 데이터를 나이브(단순)하게 독립적인 사건으로 가정하고, 이 독립 사건들을 베이즈 이론에 대입시켜 가장 높은 확률의 레이블로 분류를 실행하는 알고리즘.
- 베이즈 이론

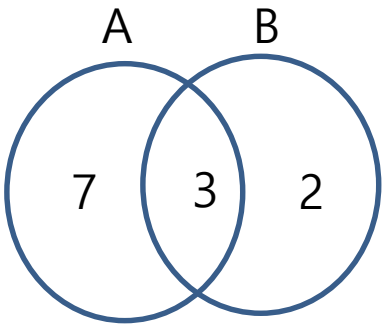
$$\mathrm{P(A \mid B)} = \frac{P(B|A)\,P(A)}{P(B)}$$

- P(A | B) : 어떤 사건 B가 일어났을 때 사건 A가 일어날 확률.
- P(B | A) : 어떤 사건 A가 일어났을 때 사건 B가 일어날 확률.
- P(A) : 어떤 사건 A가 일어날 확률.

- 조건부 확률

$$\mathrm{P(A \mid B)} = \frac{P(A \cap B)}{P(B)}$$

조건부 확률



- 사건이 발생한 횟수 : 12
- A 사건이 발생한 횟수 : 7 + 3 = 10
- B 사건이 발생한 횟수 : 3 + 2 = 5
- A 사건과 B 사건이 동시에 일어난 횟수 : 3

1. A 사건이 일어난 확률

   - P(A) = 10 / 12

2. B 사건이 일어난 확률

   - P(B) = 5 / 12

3. B 사건이 일어났을 때 A 사건이 일어날 확률

   - P(A|B) = 3 / (3 + 2) = 0.6

4. A 사건이 일어났을 때 B 사건이 일어날 확률

   - P(B|A) = 3 / (7 + 3) = 0.3

5. B 사건이 일어났을 때 A 사건이 일어날 확률을 다른 방식으로 계산하는 방법

   - P(A|B) = P(B|A) * P(A) / P(B) = 0.3 * (10/12) / (5/12) = 0.6

나이브 베이즈 알고리즘을 머신러닝에 응용하기

- A를 레이블, B를 데이터의 특징으로 대입해 보면

  - P(레이블 | 데이터 특징) = P(데이터 특징 | 레이블) * P(레이블) / P(데이터 특징)

- 즉, **어떤 데이터가 있을 때 그에 해당하는 레이블은 기존 데이터의 특징 및 레이블의 확률을 사용해 구할 수 있다는 것**.

나이브 베이즈 알고리즘의 기초 응용 사례

- 치킨 집에서 손님이 주문을 할 때 맥주를 주문할지 안 할지 예측.
- 저녁에 손님이 한 명 와서 주문 시 나이브 베이즈 공식에 따르면 손님이 맥주를 주문 할 확률.
  - P(주문|저녁) = P(저녁|주문) * P(주문) / P(저녁)

    = (3 / 4) * (4 / 10) / (5 / 10) = 0.6

시간	맥주
오전	주문 안 함
오전	주문 안 함
점심	주문함
점심	주문 안 함
점심	주문 안 함
저녁	주문함
저녁	주문함
저녁	주문함
저녁	주문 안 함
저녁	주문 안 함

- 기존 손님들의 주문 내역

나이브 베이즈 알고리즘의 고급 응용 사례

- 치킨 집에 저녁 성인 손님이 주문을 할 때 맥주를 주문할지 안 할지 예측.
- 저녁에 성인 손님이 한 명 와서 주문 시 나이브 베이즈 공식에 따르면 손님이 맥주를 주문할 확률.
  - P(주문|저녁, 성인)

    = P(저녁, 성인|주문) * P(주문) / P(저녁, 성인)
- 결합 확률 : 두 가지 이상의 사건이 동시에 발생하는 확률.
  - P(A, B) = P(A|B) P(B)

✓ 베이즈 알고리즘으로 위의 공식을 풀려고 하면 조건부 확률에 따른 많은 계산이 필요하기 때문에 나이브 베이즈 알고리즘에서는 나이브(단순)하게 모든 사건을 독립적인 사건으로 간주해 계산의 복잡성을 낮춘다.

시간	성인여부	맥주
오전	성인	주문 안 함
오전	미성년자	주문 안 함
점심	성인	주문함
점심	미성년자	주문 안 함
점심	미성년자	주문 안 함
저녁	성인	주문함
저녁	성인	주문함
저녁	성인	주문함
저녁	성인	주문 안 함
저녁	미성년자	주문 안 함

- 기존 손님들의 주문 내역

독립 사건일 경우의 조건부 확률

- 독립 사건일 경우의 조건부 확률은 다음과 같이 단순해 짐.
  - P(A, B) = P(A) * P(B)
- 나이브 베이즈 알고리즘을 통해 "저녁에 성인이 치킨 집에 와서 맥주를 주문할 확률" 계산
  - P(주문 | 저녁, 성인) = P(저녁|주문) * P(성인|주문) * P(주문) / P(저녁) * P(성인)
- 나이브 베이즈 알고리즘을 통해 "저녁에 성인이 치킨 집에 와서 맥주를 주문하지 않을 확률" 계산
  - P(주문 안함 | 저녁, 성인)

    = P(저녁|주문 안함) * P(성인|주문 안함) * P(주문 안함) / P(저녁) * P(성인)
- 여기서 실제로 필요한 것은 두 값의 대소를 비교해서 큰 값을 선택하는 것이므로 다음과 같이 공통 분모를 계산식에서 제거해서 계산량을 더 줄일 수 있음.
  - P(주문 | 저녁, 성인) = P(저녁|주문) * P(성인|주문) * P(주문) = 0.3
  - P(주문 안함 | 저녁, 성인) = P(저녁|주문 안함) * P(성인|주문 안함) * P(주문 안함) = 0.066

✓ P(주문 | 저녁, 성인)이 더 크므로 저녁에 성인이 치킨 집에 올 경우 맥주를 주문할 것으로 분류.

특징이 여러 개인 경우의 나이브 베이즈 공식

$$P(y \mid x_1, \dots, x_n) = \frac{P(x_1 \mid y)\, P(x_2 \mid y) \dots P(x_n \mid y)\, P(y)}{P(x_1) P(x_2) \dots P(x_n)}$$

- 나이브 가정에 의해 모든 특징들을 독립적인 확률로 계산하는 것을 확인할 수 있음.
- 여러 확률의 곱을 나타내는 $P(x_1 \mid y) * P(x_2 \mid y) * \cdots * P(x_n \mid y)$ 부분을 간단하게 공식화하면

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1) P(x_2) \dots P(x_n)}$$

- 모든 레이블들은 모든 특징들의 곱이라는 공통 분모를 가지고 있음.
- 레이블 확률의 대소 비교만 필요한 나이브 베이즈에서는 공통 분모를 제거해서 계산량을 줄일 수 있음.

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

- 공통 분모를 제거한 후, 가장 높은 수치를 지닌 레이블로 데이터를 분류.

$$y = argmax_y\, P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

사이킷런의 두 가지 나이브 베이즈 분류 모델

- 다항 분포 나이브 베이즈(Multinomial Naïve Bayes) 분류
  - 데이터의 특징이 출현 횟수로 표현 됐을 때 사용.
  - 예를 들어, 주사위를 10번 던졌을 때 1이 한 번, 2가 두 번, 3이 세 번, 4가 네 번 나왔을 경우, 주사위를 10번 던진 결과 데이터를 (1, 2, 3, 4, 0, 0)과 같이 나타낼 수 있다. 각 인덱스는 주사위의 면을 뜻하며, 데이터의 숫자는 출현 횟수를 나타낸 것임.
  - 이와 같이 데이터의 출현 횟수에 따라 값을 달리한 데이터에 사용.
  - 예) 영화 감상평을 토대로 긍정적/부정적 리뷰 분류.

- 베르누이 나이브 베이즈 모델(Bernoulli Naïve Bayes)
  - 데이터의 특징이 0 또는 1로 표현 됐을 때 사용.
  - 예를 들어, 주사위를 10번 던졌을 때 1이 한 번, 2가 두 번, 3이 세 번, 4가 네 번 나왔을 경우, 주사위를 10번 던진 결과 데이터를 (1, 1, 1, 1, 0, 0)과 같이 나타낼 수 있다. 각 인덱스는 주사위의 면을 뜻하며, 숫자가 출현했을 때는 1, 출현하지 않았을 때는 0으로 나타낸 것임.
  - 이와 같이 데이터 출현 여부에 따라 1 또는 0으로 구분 됐을 때 사용.
  - 예) 스팸 메일 분류

가우시안 나이브 베이즈 분류

- 앞서 다룬 예제는 데이터의 특징들이 이산적(discrete)인 간단한 예임.
- 데이터의 특징들이 연속적인 경우에는 가우시안 나이브 베이즈 분류를 사용하는 것을 추천.
- 특징들의 값들이 정규 분포(가우시안 분포)돼 있다는 가정하에 조건부 확률을 계산.
- 연속적인 성질이 있는 특징이 있는 데이터를 분류하는데 적합.
- 예) 붓꽃(iris) 데이터셋 분류

- cf) 스무딩(smoothing) 이란?
  - 이산적인 데이터의 경우 빈도수가 0인 경우가 발생
  - 예를 들어, 나이브 베이즈 기반 스팸 메일 필터를 가동시키는데, 학습 데이터에 없던 단어가 실제 상황에서 나타나게 되면 확률이 0이 되어 스팸 분류가 어려워짐.
  - 스무딩은 학습 데이터에 없던 데이터가 출현해도 빈도수에 1을 더해서 확률이 0이 되는 현상을 방지

나이브 베이즈 – 장/단점

- 장점
  - 모든 데이터의 특징이 독립적인 사건이라는 나이브 가정에도 불구하고 실전에서 높은 정확도를 보이며, 특히 문서 분류 및 스팸 메일 분류에 강한 면모를 보임.
  - 나이브 가정에 의해 계산 속도가 다른 모델들에 비해 상당히 빠름.

- 단점
  - 모든 데이터의 특징을 독립적인 사건이라고 가정하는 것은 문서 분류에 적합할 지는 모르나, 다른 분류 모델에는 제약이 될 수 있음.

대표적 머신러닝 알고리즘

- k-최근접 이웃(k-Nearest Neighbor, kNN)
- 서포트 벡터머신(Support Vector Machine, SVM)
- 의사결정 트리(Decision Tree)
- 나이브 베이즈(Naïve Bayes)
- **앙상블(Ensemble)**
- 군집화(Clustering)
- 주성분 분석(Principal Component Analysis, PCA)
- 선형 회귀(Linear Regression)
- 로지스틱 회귀(Logistic Regression)
- 딥러닝(Deep Neural Network, DNN)

앙상블(Ensemble) – 배깅(bagging)

- 여러 개의 분류 모델을 조합해서 더 나은 성능을 내는 방법.
- 배깅(bagging)
  - 한 가지 분류 모델을 여러 개 만들어서 서로 다른 학습 데이터로 학습시킨 후(부트스트랩), 동일한 테스트 데이터에 대한 서로 다른 예측값들을 투표를 통해(aggregating) 가장 높은 예측값으로 최종 결론을 내리는 앙상블 기법.
  - 어원 : 부트스트랩(bootstrap) + 어그리게이팅(aggregating)
  - 과대적합이 쉬운 모델에 상당히 적합한 앙상블.

부트스트랩(Bootstrap)

- 데이터를 조금은 편향되도록 샘플링하는 기법.
- 과대적합 모델의 특징이 학습 데이터에 대한 분산은 높고 편향은 적은 모델.
- 데이터 샘플링 시 편향을 높임으로써 분산이 높은 모델의 과대적합 위험을 줄이는 효과를 줌.
- N개의 데이터를 총 K개의 데이터로 나누어 담는다. 나누어 담을 때 중복을 허용해서 데이터의 편향을 높임.
- 편향된 데이터로 학습된 K개의 의사결정 트리는 N개의 데이터로 학습된 1개의 의사결정 트리보다 편향이 높아 과대적합될 확률이 적어짐.

예) - 전체 데이터 : [1, 2, 3, 4, 5, 6, 7, 8, 9]

- 의사결정 트리 6개를 배깅할 경우
  - 의사결정 트리1의 학습 데이터 : [1, 2, 3, 4, 5, 1]
  - 의사결정 트리2의 학습 데이터 : [2, 3, 4, 5, 1, 2]
  - 의사결정 트리3의 학습 데이터 : [3, 4, 5, 6, 7, 3]
  - 의사결정 트리4의 학습 데이터 : [4, 5, 6, 7, 8, 4]
  - 의사결정 트리5의 학습 데이터 : [5, 6, 7, 8, 9, 5]
  - 의사결정 트리6의 학습 데이터 : [6, 7, 8, 9, 1, 6]

- 각 분류 모델은 총 N개의 데이터보다 적은 데이터로 학습.
- 중복된 데이터를 허용함으로써 편향이 높은 학습 데이터로 학습.
- 데이터 샘플링 크기는 보통 전체 데이터의 60~70%를 사용.

Aggregating

- 여러 분류 모델이 예측한 값들을 조합해서 하나의 결론을 도출하는 과정.
- 결론은 투표를 통해 결정.

- 하드 보팅(Hard Voting)
  - 배깅에 포함된 K개의 분류 모델에서 최대 득표를 받은 예측값으로 결론을 도출.
  - ex) 손글씨 숫자를 보고, 숫자를 1~9로 분류하는 의사결정 트리 6개를 학습한 후, 숫자 7을 보여줬을 경우 다음과 같이 각각 분류했다고 가정.

분류모델	분류값
의사결정 트리1	1
의사결정 트리2	2
의사결정 트리3	7
의사결정 트리4	7
의사결정 트리5	7
의사결정 트리6	7

- 총 6개의 투표 중, 7이 4개의 최다 득표를 받아 하드보트 어그리게이팅은 7로 입력값을 정확히 예측.

Aggregating

- 소프트 보팅(Soft Voting)
  - 하드 보팅보다 더 정교한 투표 방식.
  - 하드 보팅 같은 경우 각 분류 모델은 최고의 확률을 갖는 분류값만을 aggregating할 때 리턴하는 반면, 소프트 보팅은 모든 분류값의 확률을 리턴함.
  - 따라서 하드 보팅은 단순히 가장 많은 투표를 받은 분류값을 단순히 aggregatin의 결론으로 도출하는 반면, 소프트 보팅은 각 분류값별 확률을 더해준 값을 점수로 사용해 최대 점수를 가진 분류값을 결론으로 도출함.
  - ex)

분류모델	1	2	3	4	5	6	7	8	9
의사결정 트리1	0.9	0.1	0	0	0	0	0	0	0
의사결정 트리2	0	0.8	0.1	0.1	0	0	0	0	0
의사결정 트리3	0	0	0.1	0	0	0	0.9	0	0
의사결정 트리4	0	0	0	0.1	0	0	0.9	0	0
의사결정 트리5	0	0	0	0	0	0	1	0	0
의사결정 트리6	0.4	0	0	0	0	0	0.6	0	0

- 숫자 7이 입력됐을 때 각 의사결정 트리의 분류값별 확률

분류값	확률값	최종점수
1	0.9+0.4	1.3
2	0.1+0.8	0.9
3	0.1+0.1	0.2
4	0.1+0.1	0.2
5	0	0
6	0	0
7	0.9+0.9+1+0.6	3.4
8	0	0
9	0	0

- 소프트 보팅의 각 분류값별 점수 계산

랜덤 포레스트(RandomForest)

- 여러 의사결정 트리를 배깅해서 예측을 실행하는 모델.
- 배깅이 모든 분류 모델에 적용 가능하지만, 특히 과대적합되기 쉬운 의사결정 트리에 적용하면 확실히 과대적합을 줄여 성능이 높아지는 혜택을 보기 때문에 배깅은 많은 의사결정 트리 모델의 개선을 이뤘고, 여러 개의 나무들이 모여 있다는 개념에서 랜덤 프레스트라는 이름이 생겨남.
- 의사결정 트리에서는 최적의 특징으로 트리를 분기하는 반면, 랜덤 포레스트는 각 노드에 주어진 데이터를 샘플링해서 일부 데이터를 제외한 채 최적의 특징을 찾아 트리를 분기함.
- 이러한 과정에서 램덤 포레스트는 또 한 번의 모델의 편향을 증가시켜 과대적합의 위험을 감소시킴.

랜덤 포레스트(RandomForest)

- 장점
  - 앙상블 효과로 의사결정 트리의 과대적합 단점을 보완한다.

- 단점
  - 조정해야 할 하이퍼 파라미터가 많다.

앙상블(Ensemble) – 부스팅(Boosting)

- 여러 개의 분류기를 만들어 투표를 통해 예측값을 결정한다는 측면에서 배깅과 동일.
- 배깅은 서로 다른 알고리즘에 기반한 여러 분류기를 병렬적으로 학습하는 반면, 부스팅은 동일한 알고리즘의 분류기를 순차적으로 학습해서 여러 개의 분류기를 만든 후, 테스트할 때 가중 투표를 통해 예측값을 결정.
- 순차적 학습과 가중 투표가 부스팅의 가장 큰 특징.

부스팅 – 순차적 학습

- 가령, 인물 사진 안에 있는 인물을 보고, 남자 또는 여자로 분류하는 의사결정 트리를 부스팅할 경우, 먼저 첫 번째 의사결정 트리를 학습함.
- 테스트 결과, 남자 분류가 미흡할 경우, 남자 학습 데이터를 보강한 후 두 번째 의사결정 트리를 학습함.
- 그리고 두 번째 의사결정 트리의 테스트 결과에 따라 학습 데이터를 보강해서 세 번째 의사결정 트리를 학습함.
- 이처럼 부스팅은 순차적으로 학습 데이터를 보강하며 동일한 알고리즘의 분류기를 여러 개 만드는 과정을 가짐.

부스팅 – 가중 투표

- 배깅은 마치 우리가 선거를 하듯, 동일한 한 표씩 부여되는 반면 부스팅은 가중 투표가 진행됨.
- 가중 투표는 투표자의 능력치에 따라 한 표의 가치가 다른 투표임.
- 팀회식 장소를 결정하는데, 사원 5명이 클럽에 가자고 했지만 부장님 3명이 삼겹살을 먹으러 가자고 했을 때, 삼겹살로 회식이 결정되는 것과 같음.
- 3가지 분류기의 검증 결과, 정확도가 다음과 같다고 가정하면

	분류기1	분류기2	분류기3
정확도	0.4	0.5	0.95

- 세 분류기의 정확도

	분류기1	분류기2	분류기3
분류값	남자	남자	여자

- 세 분류기의 분류값(하드보팅)

- 이 경우 가중 투표는 다음과 같이 진행.
  - 하드 보팅
    - ✓ 남자 : 0.4 + 0.5, 여자 : 0.95
    - ✓ 따라서 두 분류기가 남자라고 예측했음에도 가중 투표 결과, 사진의 인물을 여자로 분류.
  - 소프트 보팅
    - ✓ 남자: 0.4*0.7 + 0.5*0.8 + 0.95*0.1 = 0.775
    - ✓ 여자: 0.4*0.3 + 0.5*0.2 + 0.95*0.9 = 1.075.
    - ✓ 소프트 보팅의 결과, 여자로 최종 결론 도출.

	분류기1	분류기2	분류기3
분류값	남자: 0.7 여자: 0.3	남자: 0.8 여자: 0.2	남자: 0.1 여자: 0.9

- 세 분류기의 분류값(소프트보팅)

대표적 머신러닝 알고리즘

- k-최근접 이웃(k-Nearest Neighbor, kNN)
- 서포트 벡터머신(Support Vector Machine, SVM)
- 의사결정 트리(Decision Tree)
- 나이브 베이즈(Naïve Bayes)
- 앙상블(Ensemble)
- **군집화(Clustering)**
- 주성분 분석(Principal Component Analysis, PCA)
- 선형 회귀(Linear Regression)
- 로지스틱 회귀(Logistic Regression)
- 딥러닝(Deep Neural Network, DNN)

군집화(Clustering)

- **비지도학습**의 일종으로, 데이터의 특징만으로 비슷한 데이터들끼리 모아 군집된 클래스로 분류함.

- K 평균 알고리즘 – 간단하면서도 강력한 군집화 알고리즘.

- K 평균 알고리즘의 진행 순서
  1. 데이터 준비
  2. 몇 개의 클래스로 분류할 것인지 설정
  3. 클러스터의 최초 중심 설정
  4. 데이터를 가장 가까운 클러스터로 지정
  5. 클러스터 중심을 클러스터에 속한 데이터들의 가운데 위치로 변경
  6. 클러스터 중심이 바뀌지 않을 때까지 4번부터 5번 과정을 반복적으로 수행

k 평균 알고리즘 – (1) 데이터 준비

- k 평균 알고리즘은 데이터 간의 거리를 사용해 가까운 거리에 있는 데이터끼리 하나의 클래스로 묶는 알고리즘.
- 거리를 계산하기 위해 데이터는 수치화된 데이터여야 함.
- 이해를 돕기 위해 한 교실에 있는 학생들의 키와 몸무게 값으로 세 개의 군집으로 분류하는 예제를 살펴봄.

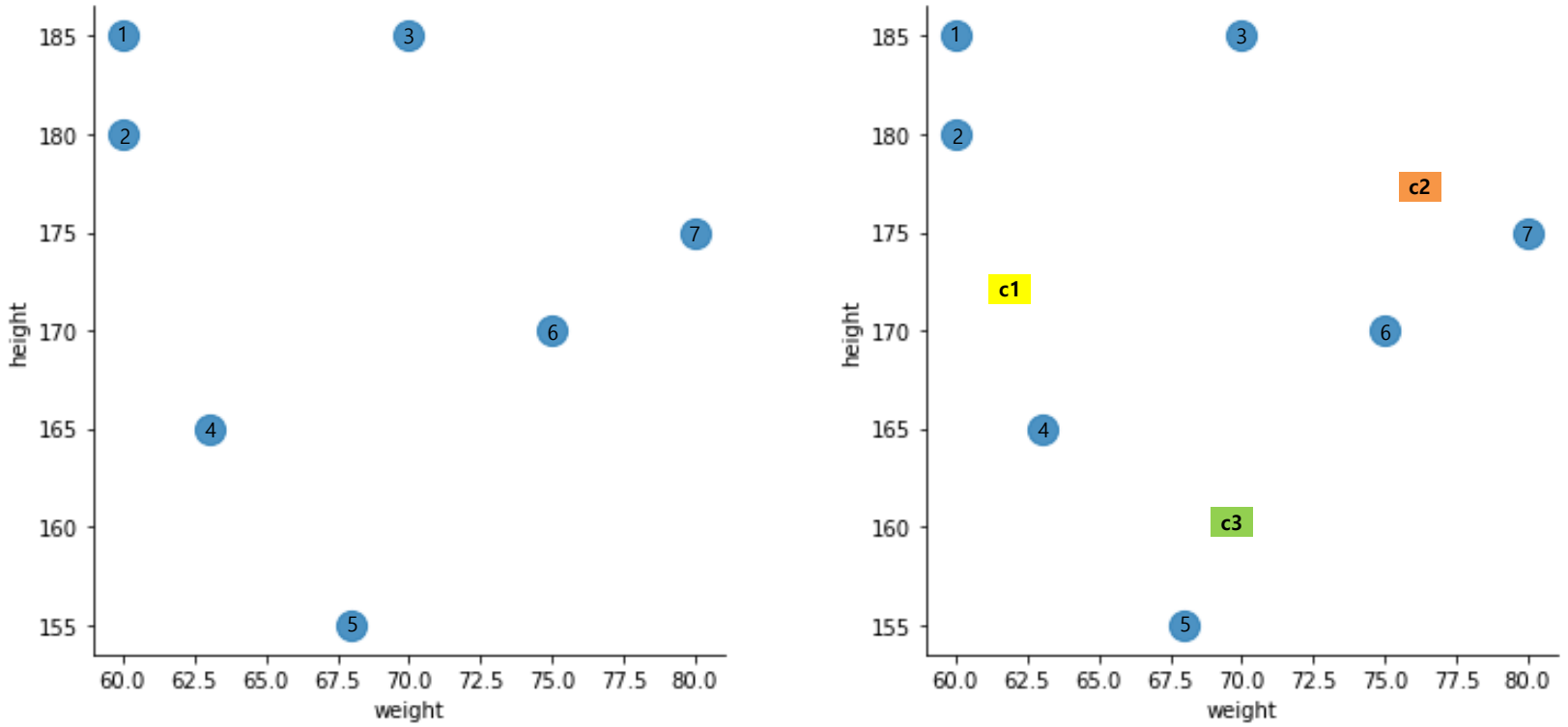
학생	키	몸무게
1	185	60
2	180	60
3	185	70
4	165	63
5	155	68
6	170	75
7	175	80

k 평균 알고리즘

- (2) 몇 개의 클래스로 분류할 것인지 정하기.
  - k 평균 알고리즘의 k는 몇 개의 클래스로 분류할 것인지를 나타내는 변수.
  - 이번 예제에서는 k를 3으로 설정해 학생들을 세 개의 집단으로 분류.
- (3) 클러스터의 최초 중심 설정.
  - k 평균의 표준 알고리즘은 클러스터의 최초 중심을 무작위로 설정함.
  - 경우에 따라 최초 중심을 k 평균 모델에 부여할 수도 있음.
    - 예를 들어, 서울의 각 지역구별 위도/경도 데이터만 있을 때 강동, 강서, 강남, 강북으로 군집화하기 위해 가장 좋은 방법은 강동, 강서, 강남, 강북의 정가운데에 있는 지역구를 최초 클러스터의 중심으로 부여하는 것일 것.
  - 사이킷런의 k 평균 라이브러리
    - 기본적으로 kmean++라는 알고리즘을 써서 클러스터의 최초 중심을 설정함.
    - kmean++는 최초 데이터 포인트 지점을 첫 번째 클러스터의 중심으로 설정하고, 최초 데이터 포인트에서 가장 먼 데이터 포인트를 두 번째 클러스터의 중심으로, 그리고 기존의 클러스터 중심에서 가장 먼 데이터 포인트를 그 다음 클러스터의 중심으로 설정.
    - 표준 k 평균 알고리즘의 랜덤 중심 설정은 때로는 초기 중심이 한 군데로 집약되어 군집하는 데 시간이 오래 걸리고, 군집의 결과가 나쁠 경우가 더러 있어 단점을 보완하고자 만들어진 알고리즘.

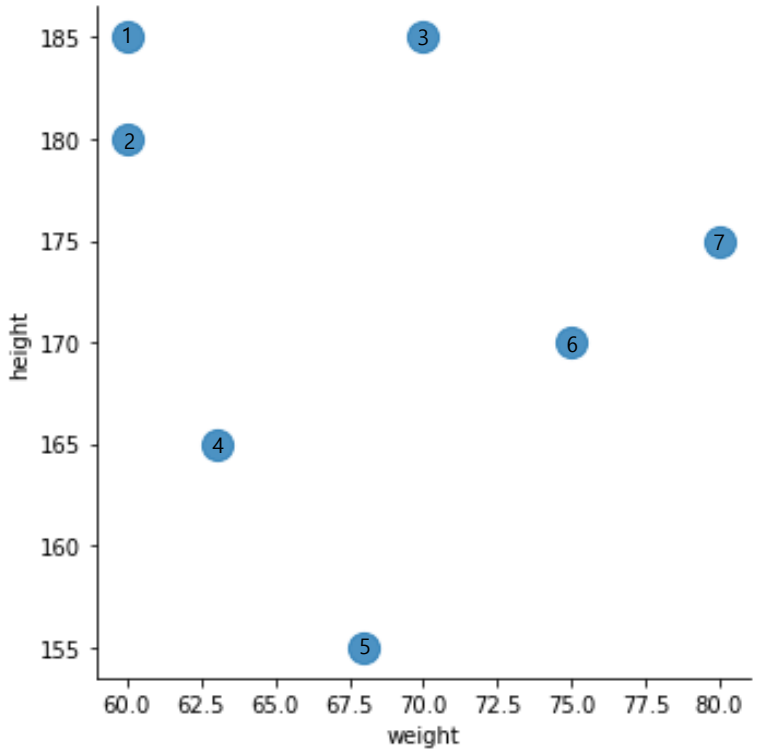
k 평균 알고리즘

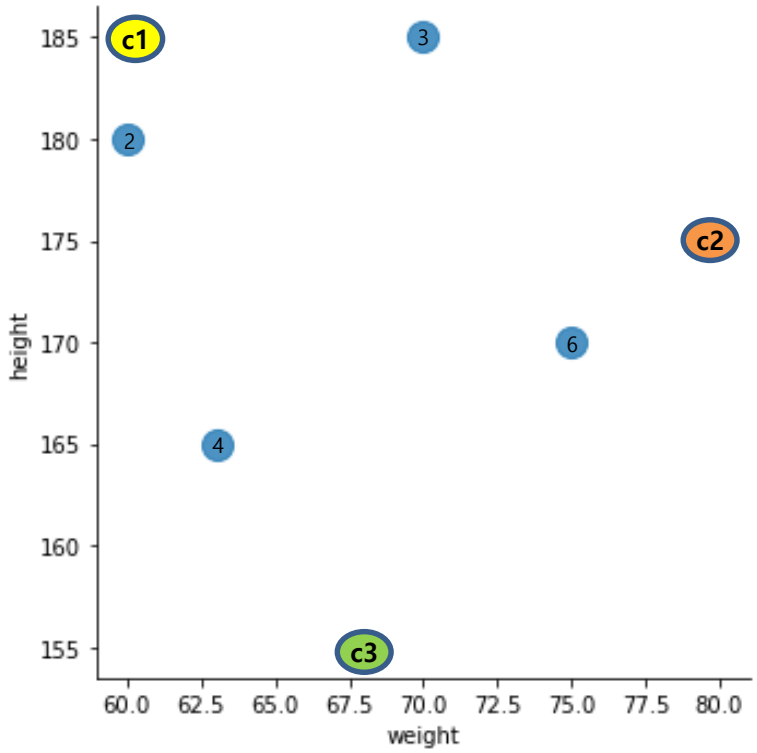
- 클러스터의 최초 중심 설정 – 무작위로 클러스터 중심 정하기.



k 평균 알고리즘

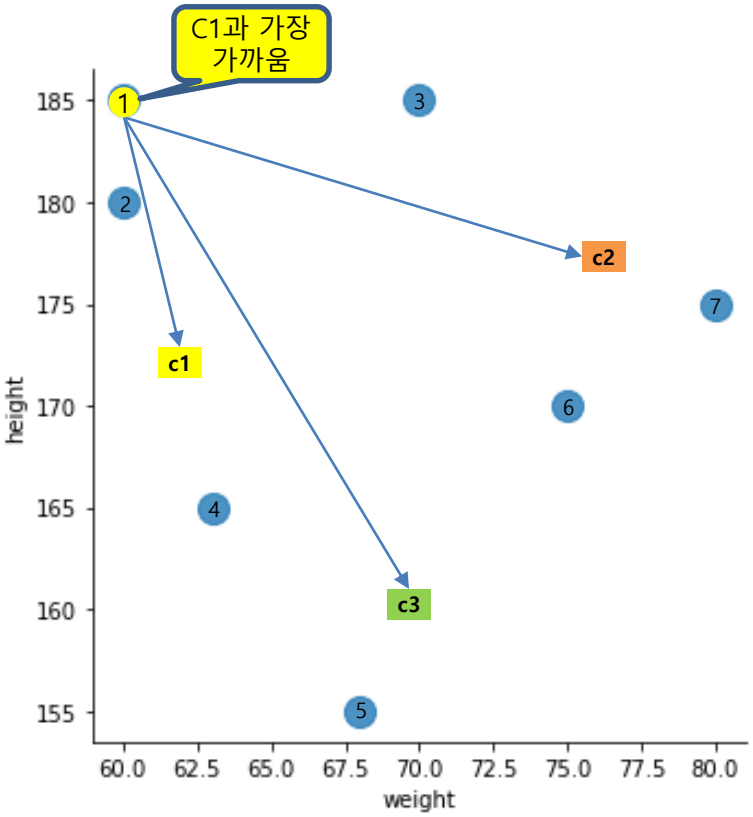
- 클러스터의 최초 중심 설정 – kmean++의 중심 정하기.

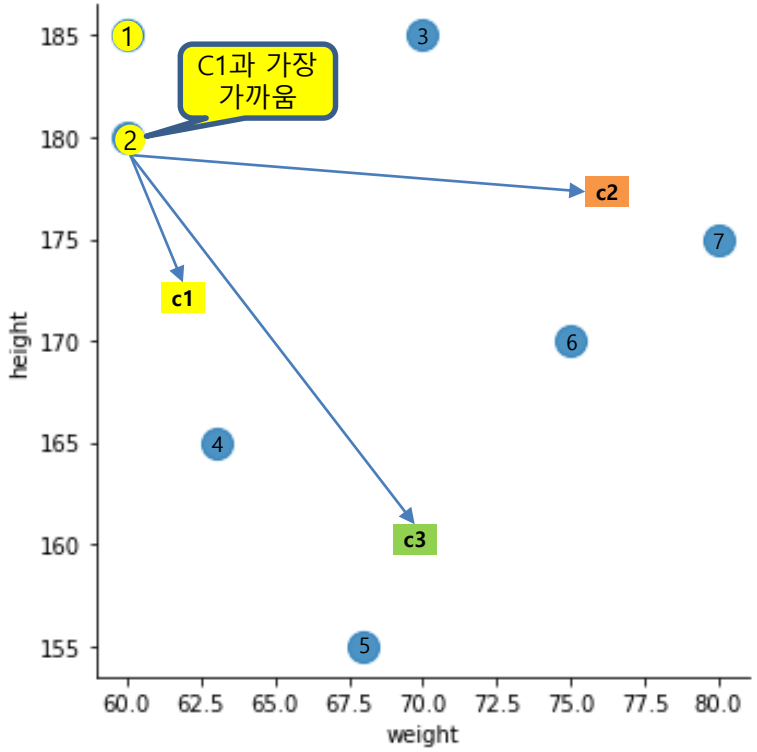




k 평균 알고리즘

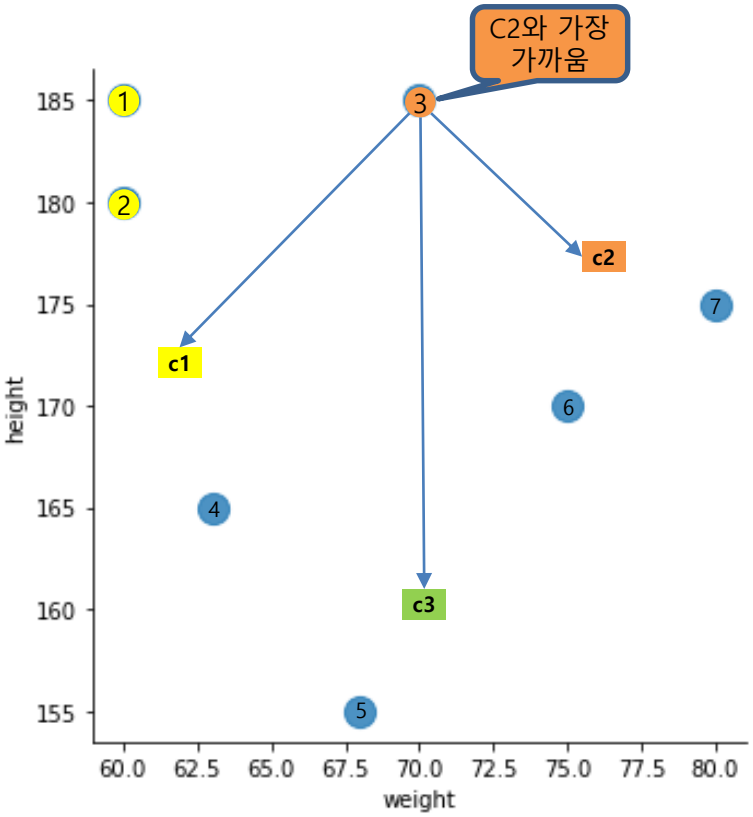
- 표준 k 평균 알고리즘에 따라 무작위로 클러스터의 중심을 설정하고 각 (4) **데이터를 가장 가까운 클러스터에 지정**하는 과정.

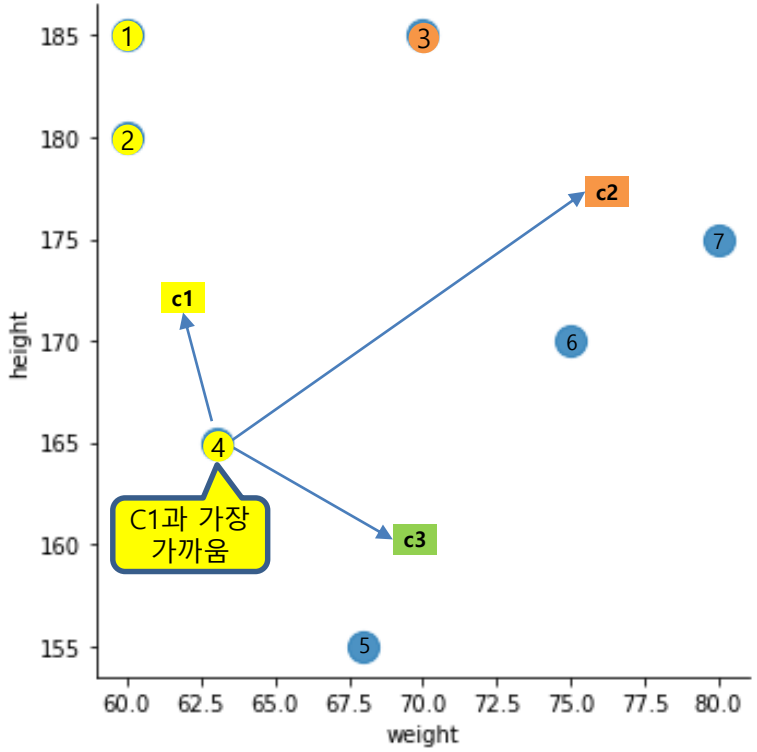




k 평균 알고리즘

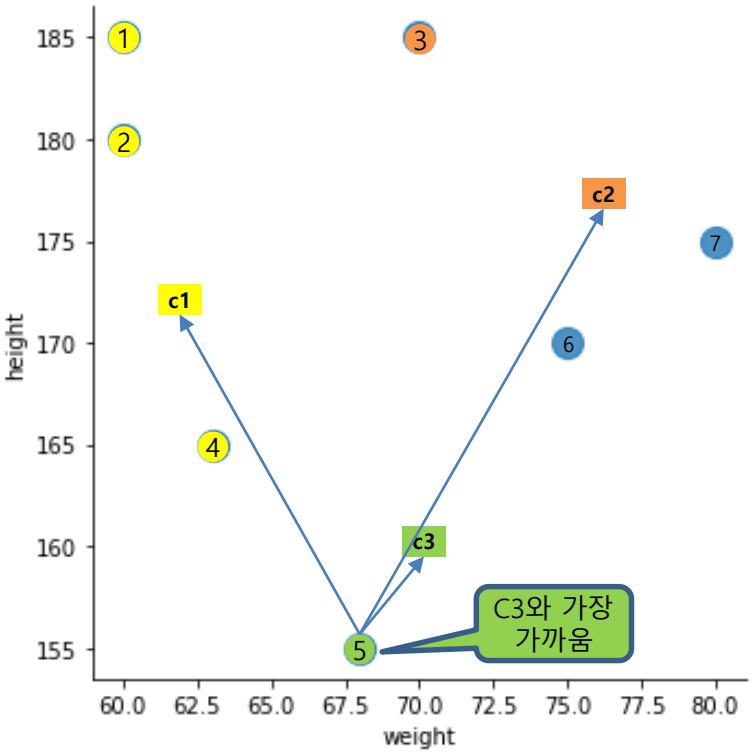
- 표준 k 평균 알고리즘에 따라 무작위로 클러스터의 중심을 설정하고 각 (4) **데이터를 가장 가까운 클러스터에 지정**하는 과정.

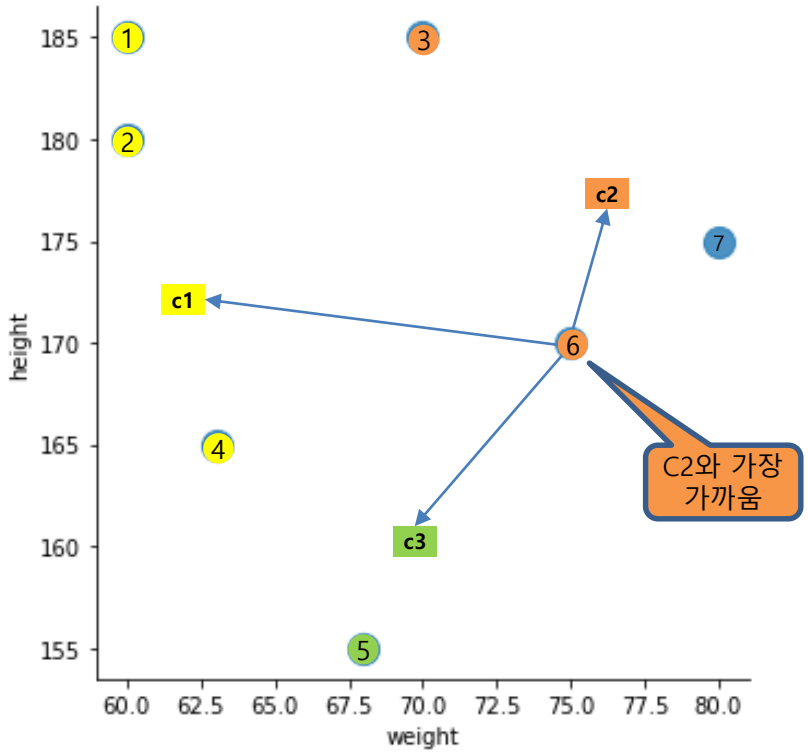




k 평균 알고리즘

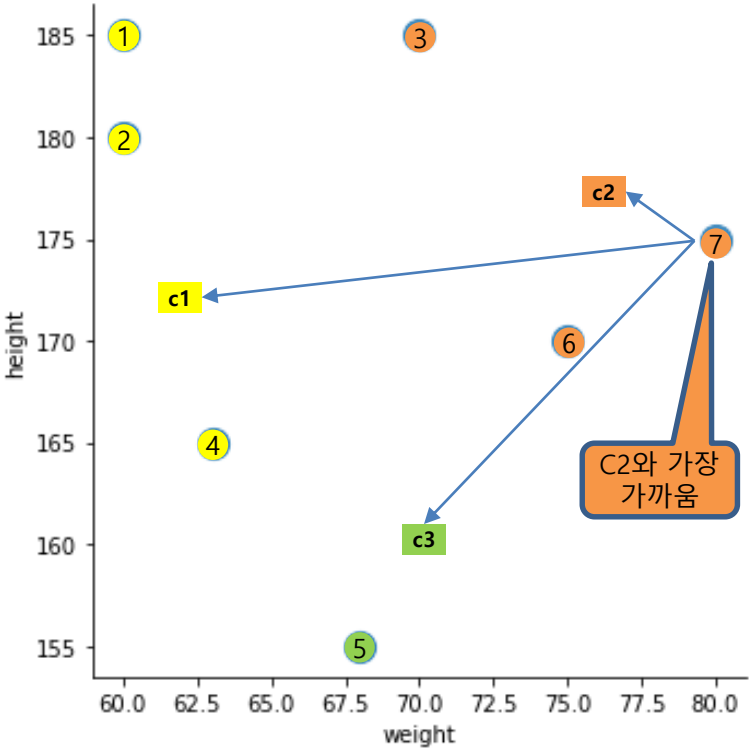
- 표준 k 평균 알고리즘에 따라 무작위로 클러스터의 중심을 설정하고 각 (4) **데이터를 가장 가까운 클러스터에 지정**하는 과정.

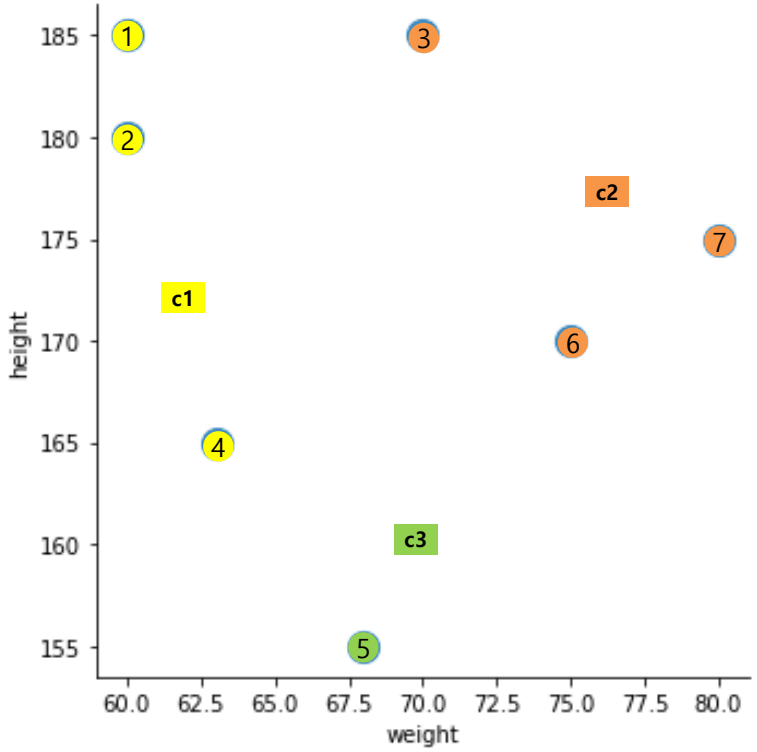




k 평균 알고리즘

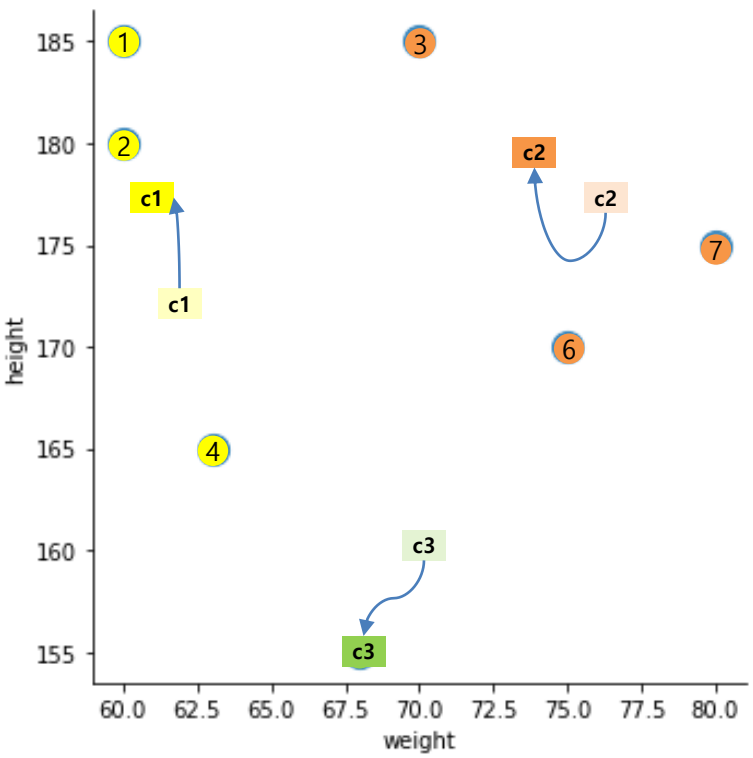
- 표준 k 평균 알고리즘에 따라 무작위로 클러스터의 중심을 설정하고 각 (4) **데이터를 가장 가까운 클러스터에 지정**하는 과정.

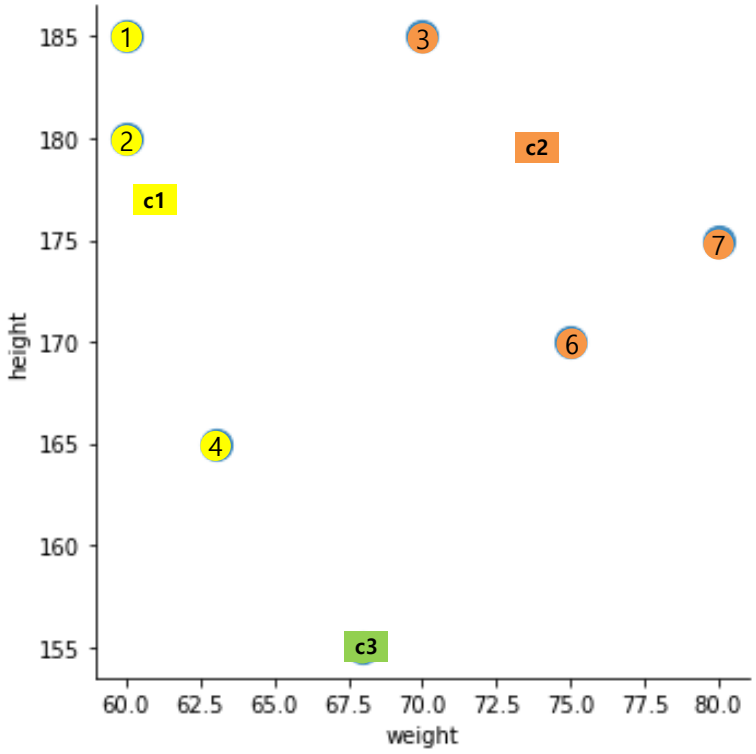




k 평균 알고리즘

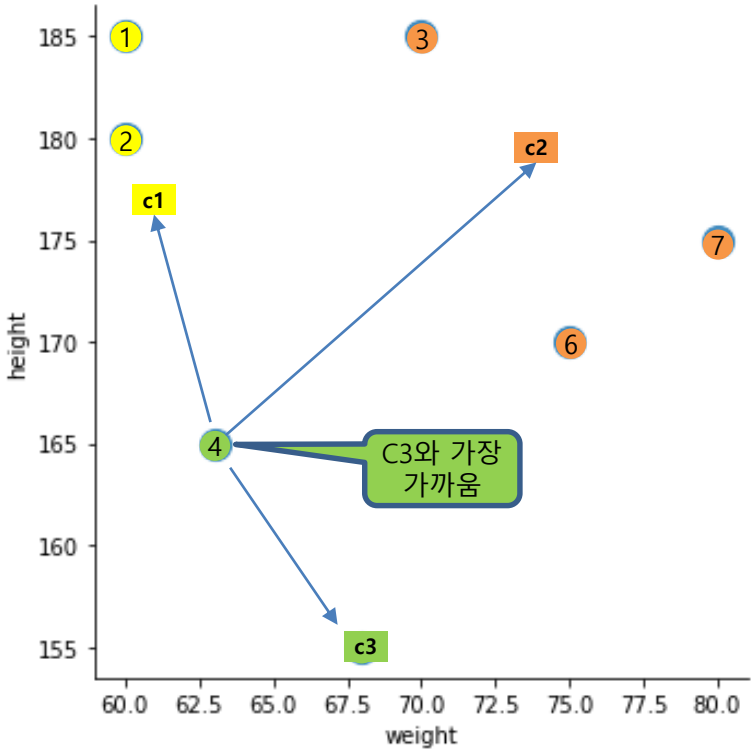
- 데이터 순회가 완료되면 각 (5) **클러스터 중심을 클러스터에 속한 데이터들의 가운데 위치로 변경**.

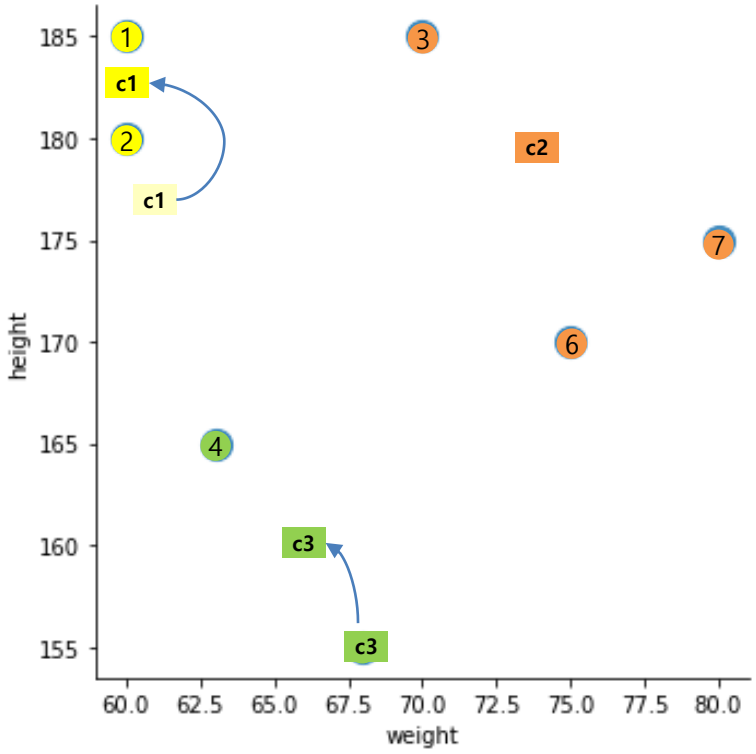




k 평균 알고리즘

- 클러스터의 중심이 바뀌면 다시 첫 번째 데이터부터 마지막 데이터까지 순회하며, 데이터를 가장 가까운 클러스터로 소속시키며, (6) **클러스터 중심이 바뀌지 않을 때까지 (4)번부터 (5)번 과정을 반복 수행**.





대표적 머신러닝 알고리즘

- k-최근접 이웃(k-Nearest Neighbor, kNN)
- 서포트 벡터머신(Support Vector Machine, SVM)
- 의사결정 트리(Decision Tree)
- 나이브 베이즈(Naïve Bayes)
- 앙상블(Ensemble)
- 군집화(Clustering)
- **주성분 분석(Principal Component Analysis, PCA)**
- 선형 회귀(Linear Regression)
- 로지스틱 회귀(Logistic Regression)
- 딥러닝(Deep Neural Network, DNN)

주성분 분석(Principal Component Analysis, PCA)

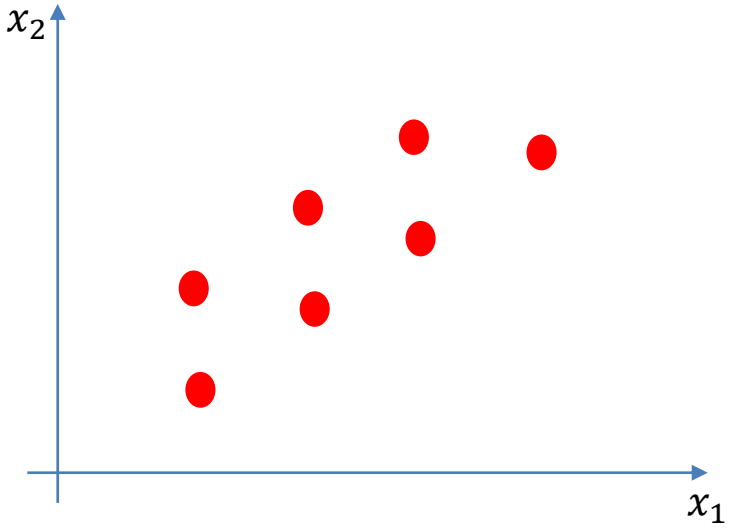
- 고차원의 데이터를 저차원의 데이터로 차원 축소하는 알고리즘.
- 주로 고차원의 데이터를 3차원 이하의 데이터로 바꿔서 시각화하는 데 많이 사용.
- 유용한 정보만 살려서 적은 메모리에 저장하거나 데이터의 노이즈를 줄이고 싶을 때도 사용.

- 주성분 분석의 특징
  - 데이터의 분산을 최대한 유지하면서 저차원으로 데이터를 변환하는데 있음.
  - 분산을 유지하는 이유는 데이터의 고유한 특성을 최대한 유지하기 위함.

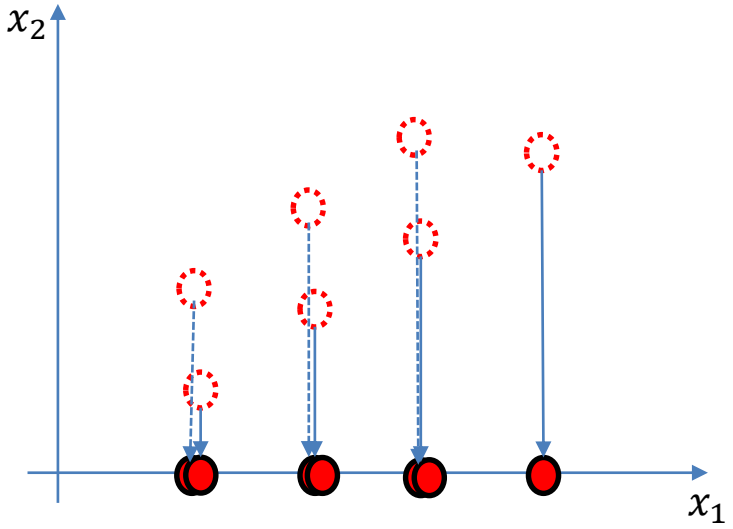
주성분 분석(Principal Component Analysis, PCA)

- 이해를 위해 쉽게 시각화 가능한 2차원 데이터를 1차원 데이터로 축소하는 과정을 살펴봄.

- 단순히 2차원 데이터를 x_1축으로 옮긴다면 다음과 같을 것.



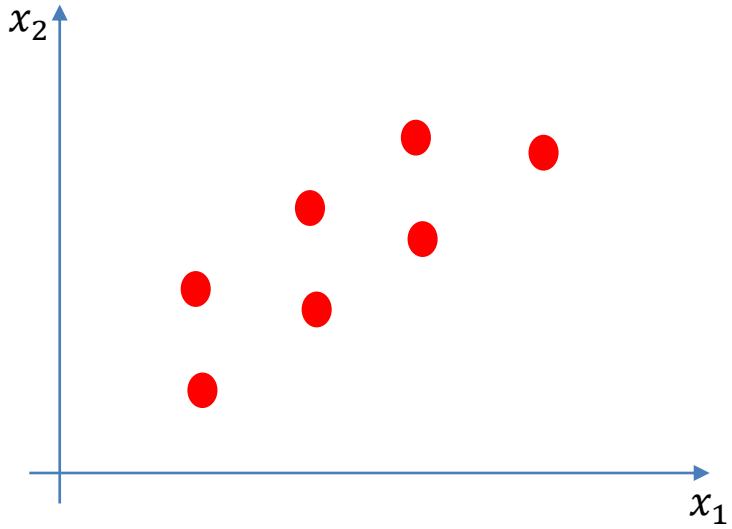
- 2차원 데이터포인트 분포도



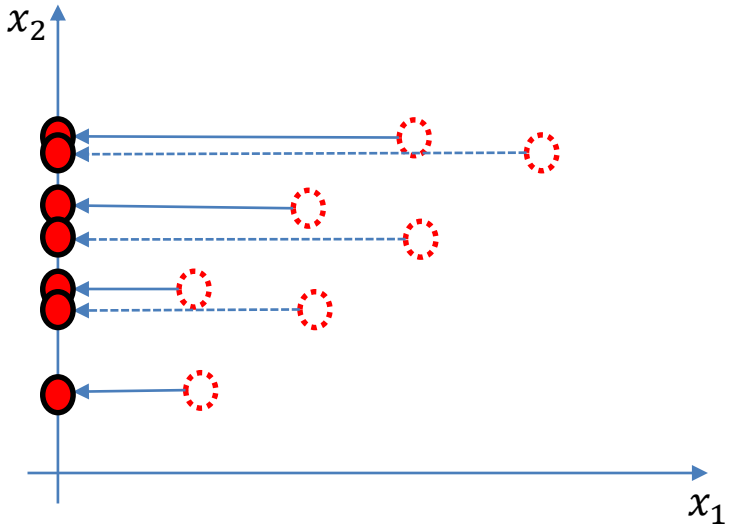
- 2차원 데이터를 x_1축에 투영시켰을 때의 시각화

주성분 분석(Principal Component Analysis, PCA)

- 단순히 2차원 데이터를 x_2축으로 옮긴다면 다음과 같을 것.



- 2차원 데이터포인트 분포도

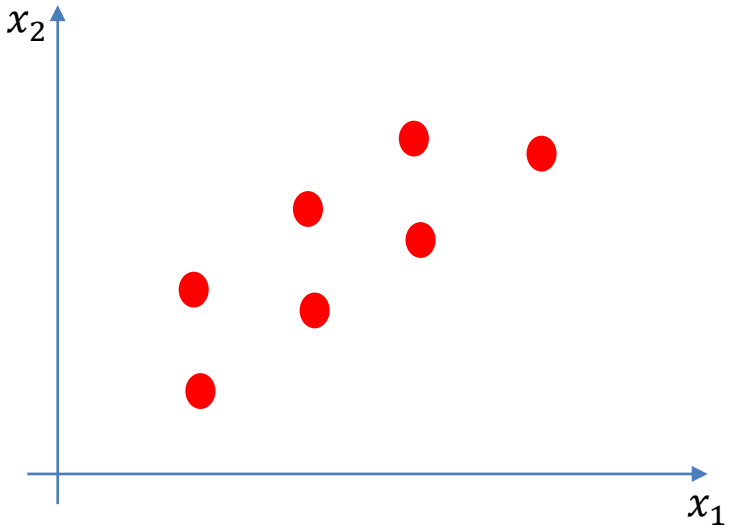


- 2차원 데이터를 x_2축에 투영시켰을 때의 시각화

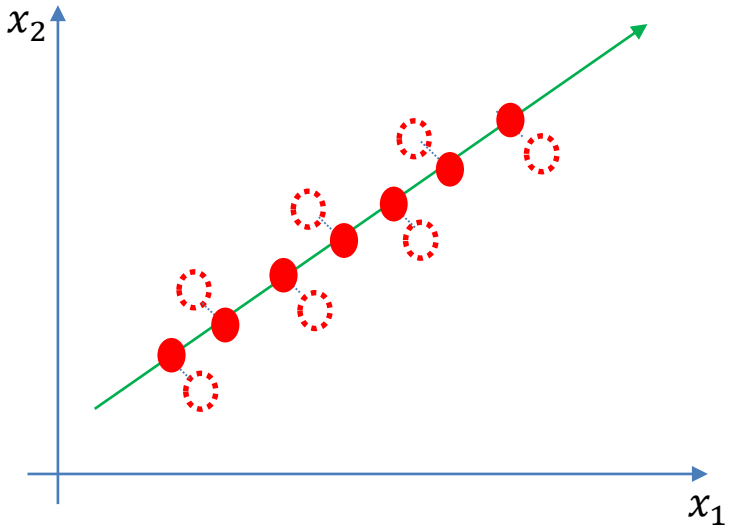
- 2차원 데이터를 1차원으로 옮겼다는 점에서는 성공적임.
- 2차원 상에 있었을 때 확실하게 구분됐던 7개의 점이, 1차원 직선상에서 많이 중첩되어 데이터를 구분하기가 어려워진 것을 눈으로 확인할 수 있음.

주성분 분석(Principal Component Analysis, PCA)

- 7개의 점들이 확실하게 구분된 1차원 직선.



- 2차원 데이터포인트 분포도

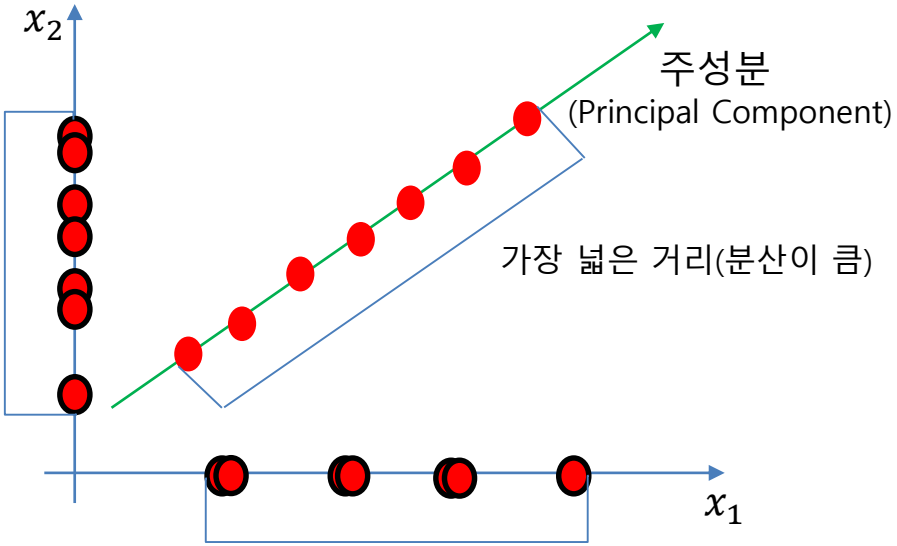


- 투영 시 정보의 유실이 없는 1차원 직선

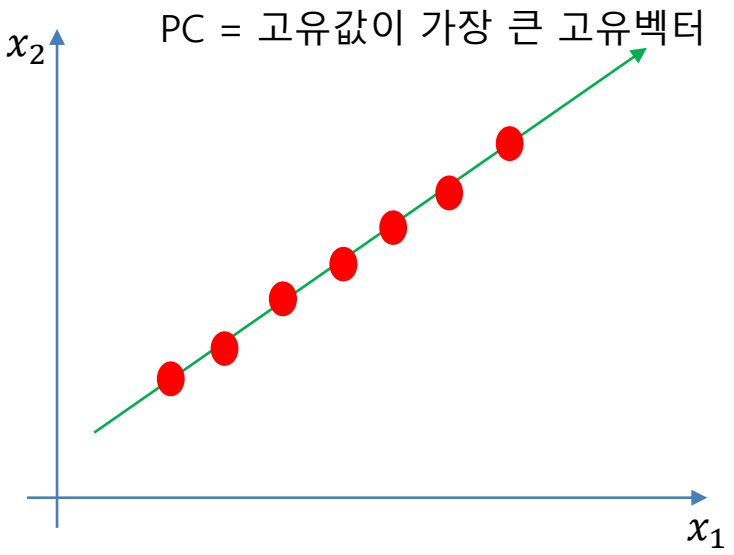
- 1차원 상에 점들이 있음에도 데이터가 하나도 중첩되지 않고 육안으로 7개의 점을 확인할 수 있음.
- 정보 이론 측면에서 해석해 본다면 고차원 데이터를 저차원 데이터로 옮겼음에도 유용한 정보의 유실이 가장 적다고 판단할 수 있음.

주성분 분석(Principal Component Analysis, PCA)

- 주성분 분석 알고리즘은 수학적인 방법으로 데이터 정보의 유실이 가장 적은 라인을 찾아냄.
- 수학적으로 '데이터의 중첩이 가장 적다 ' 라는 말은 '데이터의 분산이 가장 크다 ' 라는 말과 동일.
- 주성분 분석은 아래 그램과 같이 분산이 가장 큰 차원을 선택해 고차원 데이터를 저차원으로 축소함.
- 분산이 가장 큰 차원은 수학적으로 공분산 행렬(covariance matrix)에서 고유값(eigen value)이 가장 큰 고유 벡터(eigen vector)임.



- 각 직선별 분산 비교



- 고유 벡터 시각화

- 데이터가 5차원 데이터이고, 2차원으로 데이터의 차원을 줄이고 싶다면 주성분 분석 알고리즘은 공분산 행렬에서 고유값이 큰 순서대로 고유벡터를 정렬한 후, 가장 큰 고유 벡터와 두 번째로 큰 고유 벡터를 축으로 2차원 데이터를 만들게 됨.