# 인공지능과 신약개발을 위한 파이썬

## 5주차 머신 러닝의 이해

홍 성 은

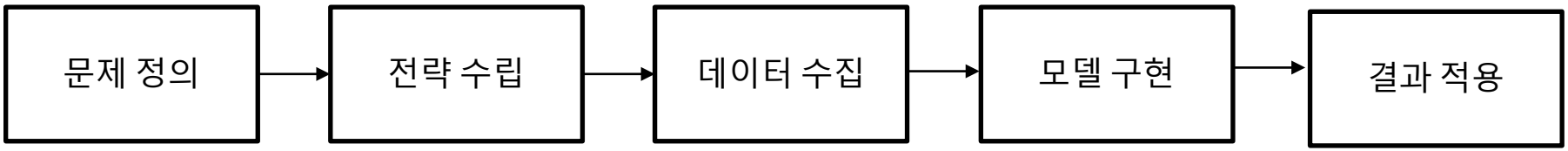
sungkenh@gmail.com

# 목차

- 데이터 분석
- 머신러닝
- 회귀분석
- 분류
- 클러스터링

# 데이터 분석

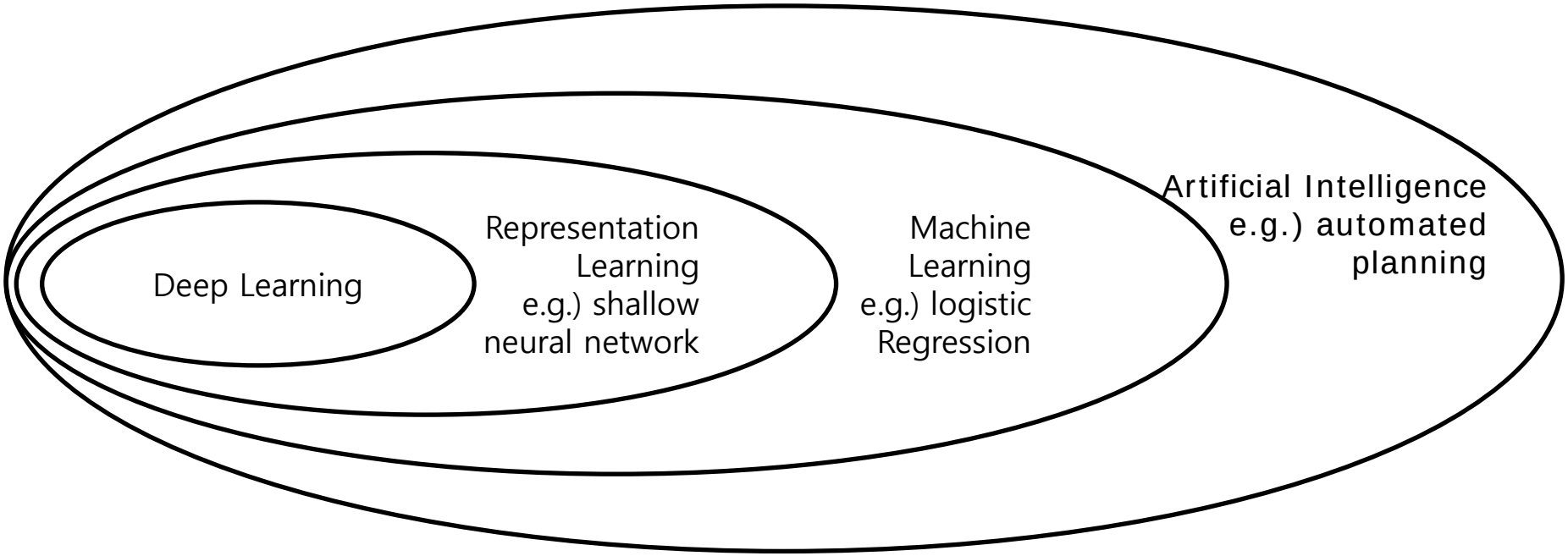
- 데이터 분석 프로세스
  - 문제 정의 : 해결하려는 문제를 명확히 정의하는 것
  - 전략 수립 : 문제 해결을 위해 어떤 데이터를 어떻게 사용할지를 정함
  - 데이터 수집 : 머신러닝에 필요한 데이터를 수집하는 것
  - 모델 구현 : 분류, 회귀, 설명, 추천 등을 위한 머신러닝 모델을 구현
  - 결과 적용 : 머신러닝 모델을 실제 상황에 적용하고 성능을 개선하는 것



# 데이터 분석

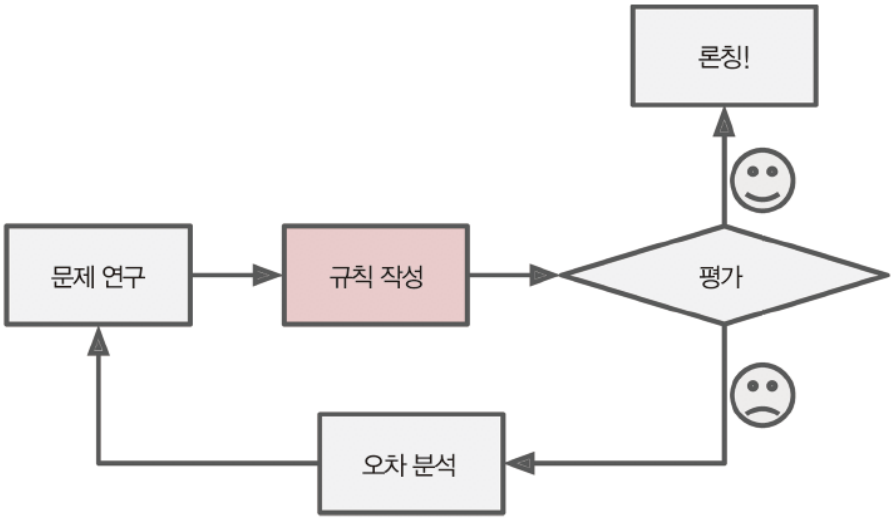
- 데이터 수집
  - 전체 과정에서 70~80%의 시간을 소모함
  - 핵심 데이터를 확보했는지 여부
  - 데이터 품질
  - 잘못된 데이터 사용은 잘못된 결과를 도출
  - 수집 가능 여부 (보유 기관의 데이터 정책)
  - 수집 주기 (일회성, 한시간/하루/한달에 한번 등)
  - 비용 (무료 또는 유상, 통신 비용 등)
  - 데이터 포맷 변경, 호환성, 처리 비용
  - 정답 데이터 셋 확보 여부
  - 수집 아이디어...

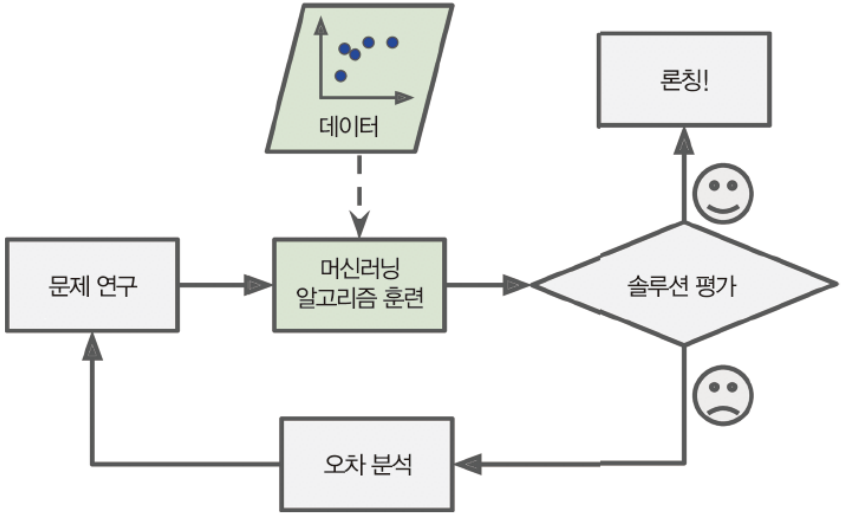
# 머신 러닝


Deep Learning
Representation Learning e.g.) shallow neural network
Machine Learning e.g.) logistic Regression
Artificial Intelligence e.g.) automated planning

# 머신 러닝

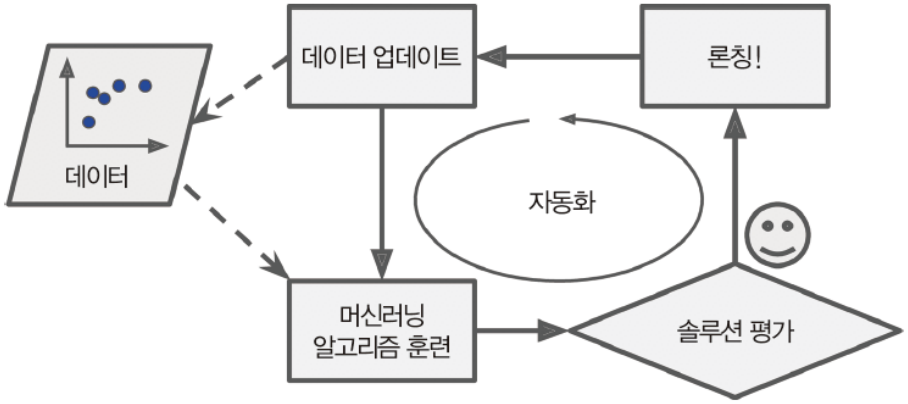
- 머신러닝은 데이터에서부터 학습하도록 컴퓨터를 프로그래밍하는 과학
- 명시적 프로그래밍 없이 컴퓨터가 학습하는 능력을 갖추게 하는 연구분야
- 딥러닝: 신경망을 기반으로 하는 머신러닝 기술
  - 마치 사람이 많은 정보에 접하면서 학습하듯이 컴퓨터도 데이터를 보고 학습하는 방법
  - 음성인식, 자동차 번호판 인식, 언어 번역, 채팅 대화, 글쓰기, 작곡 등 여러 분야에서 좋은 성과를 냄
- 스팸필터
  - 스팸메일과 일반 메일의 샘플을 이용해 스팸 메일 구분법을 학습하는 머신러닝 프로그램
  - 시스템이 학습하는데 사용하는 데이터 샘플을 **훈련 데이터**, 각 훈련 데이터를 **훈련 사례**(**샘플**)
  - 정확히 분류된 메일의 비율의 성능을 **정확도**라고 **함**



# 머신 러닝

- 머신 러닝 작업 흐름




- 머신 러닝이 유용한 문제
  - 음성인식
    - One, Two를 구분하는 프로그램을 작성한다고 했을 경우
    - Two는 높은 피치의 T로 시작하기에 높은 피치의 강도를 측정하는 알고리즘을 하드코딩해서 두개를 구분할 수 있음
    - 소음이 있는 환경에서 수백만 명이 말하는 여러 언어로 된 수천개의 단어를 구분하는 것으로 확장하기 어려움
    - 각 단어를 녹음한 샘플을 사용해 스스로 학습하는 알고리즘을 작성하는 것이 가장 좋은 솔루션

# 머신 러닝

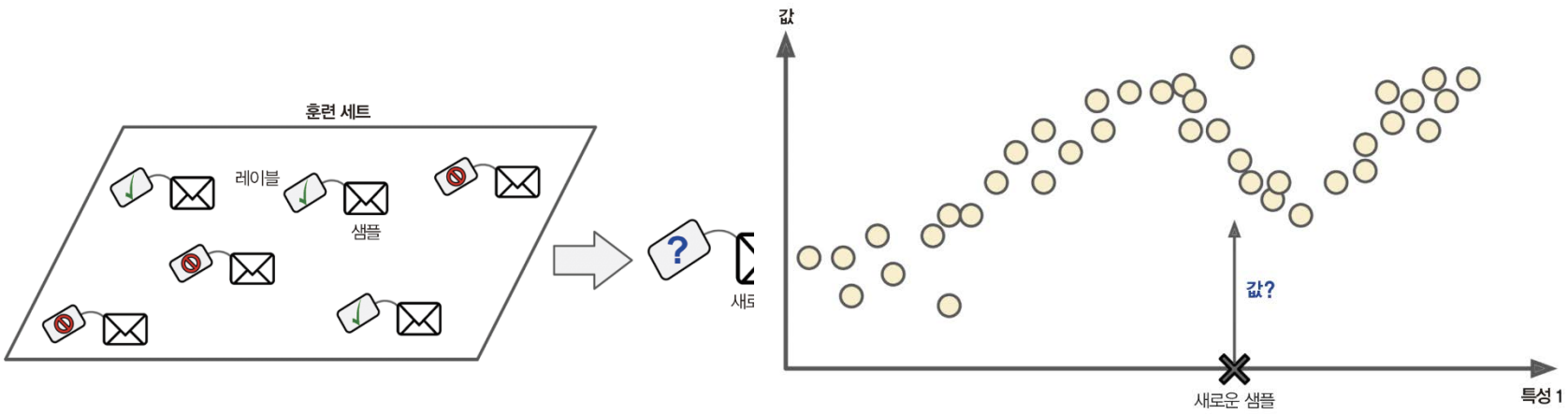
- 머신 러닝 해석
  - 머신 러닝이 학습한 것을 조사할 수 있음
  - 스팸 필터가 충분한 스팸 메일로 훈련되었다면 스팸을 예측하는데 가장 좋은 단어 및 단어의 조합을 확인할 수 있음
  - 간혹, 예상치 못한 연관성이나 새로운 추세가 발견되기도 해서 해당 문제를 더 잘 이해하도록 도와 줌
  - 머신 러닝 기술을 적용해서 대용량의 데이터를 분석하면 **겉으로 보이지 않는 패턴을 발견**할 수 있도록 해주는데 이를 **데이터 마이닝**이라고 함
- 머신 러닝 응용 분야
  - 제품 이미지를 보고 자동으로 분류하기
  - 자동으로 뉴스 기사를 분류하기
  - 내년도 회사의 수익을 예측하기
  - 음성을 듣고 이해하는 앱을 만들기
  - 구매 이력을 기반으로 고객을 나누기

# 머신 러닝

- 머신 러닝의 유형
  - 사람의 감독하에 훈련하는 것인지(지도, 비지도, 준지도, 강화학습)
  - 실시간으로 점진적인 학습을 하는지 아닌지(온라인 학습과 배치학습)
  - 단순하게 알고 있는 데이터 포인트와 새 데이터 포인트를 비교하는 것인지 훈련 데이터셋의 패턴을 발견하여 예측 모델을 만드는 것인지(사례 기반 학습, 모델 기반 학습)

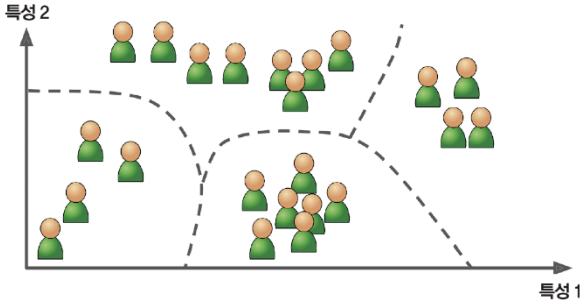
# 머신 러닝

- 지도학습
  - 지도학습은 정답이 주어지고 정답을 예측하는데 사용
  - 정답은 목적(target) 변수, 레이블이라고도 함
  - 예측은 분류와 회귀로 나누어짐
  - 분류
    - 분류(classification)란 어떤 항목(item)이 어느 그룹에 속하는지를 판별하는 기능을 말함
    - 두 가지 카테고리를 나누는 작업을 이진 분류(binary classification)라고 하고 세 개 이상의 클래스를 나누는 작업을 다중 분류(multiclass classification)라고 함
  - 회귀
    - 수치를 예측하는 것을 회귀라고 한다.

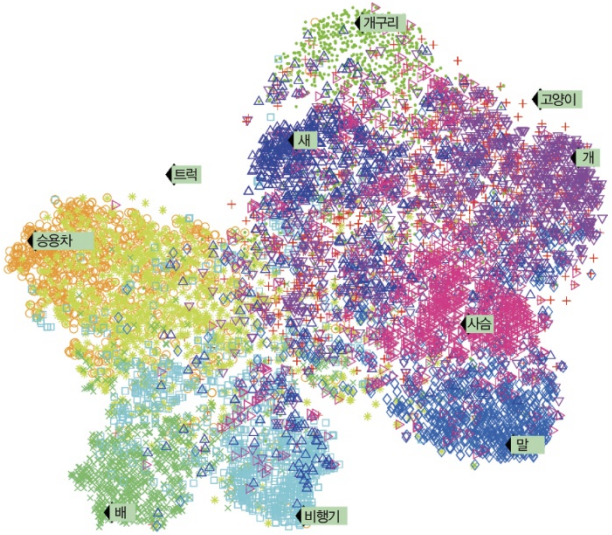


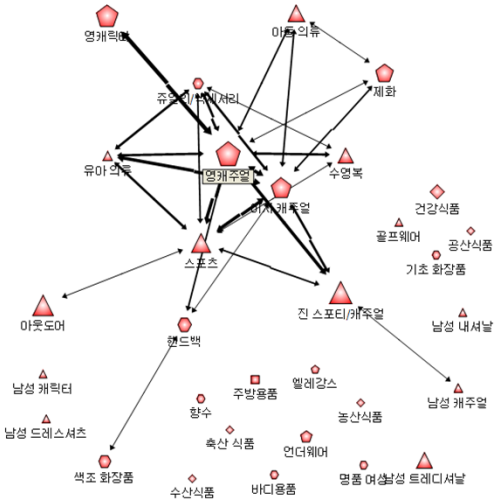
# 머신 러닝

- 비지도 학습
  - 비지도 학습이란 정답이 없이 데이터로부터 중요한 의미를 찾아내는 머신 러닝 기법임
    - 군집화: 유사한 항목들을 같은 그룹으로 묶음
    - 차원 축소 및 시각화 : 머신 러닝에 사용할 특성의 수를 줄임
    - 연관 분석
      - 어떤 사건이 다른 사건과 얼마나 자주 동시에 발생하는지 파악
      - 자주 발생하는 패턴 찾기(상품의 연관성, 취향의 연관성 등 분석)
      - 같이 구매한 상품 분석(market basket analysis, 장바구니 분석)
      - 상품의 진열 배치 및 상품 프로모션(쿠폰 발행 등)에 활용



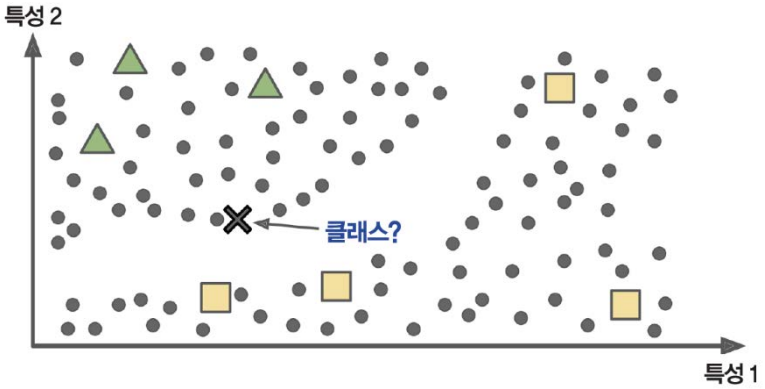




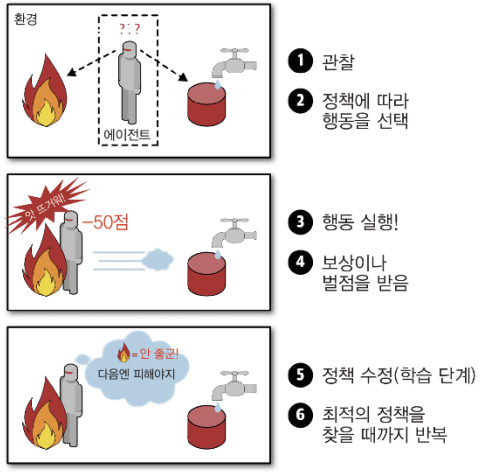



# 머신 러닝

- 준지도 학습
  - 데이터에 레이블을 다는 것이 시간과 비용이 많이 필요하기 때문에 레이블이 없는 샘플이 많고 레이블이 있는 샘플이 적음
  - 정답이 일부만 있는 경우를 준지도 학습이라고 함




- 강화 학습
  - 학습하는 시스템을 **에이전트**
  - 환경을 관찰해서 행동을 실행하고 그 결과로 **보상, 벌점**부과
  - 시간이 지나며 가장 큰 보상을 얻기 위해 **정책**이라는 전략을 스스로 학습

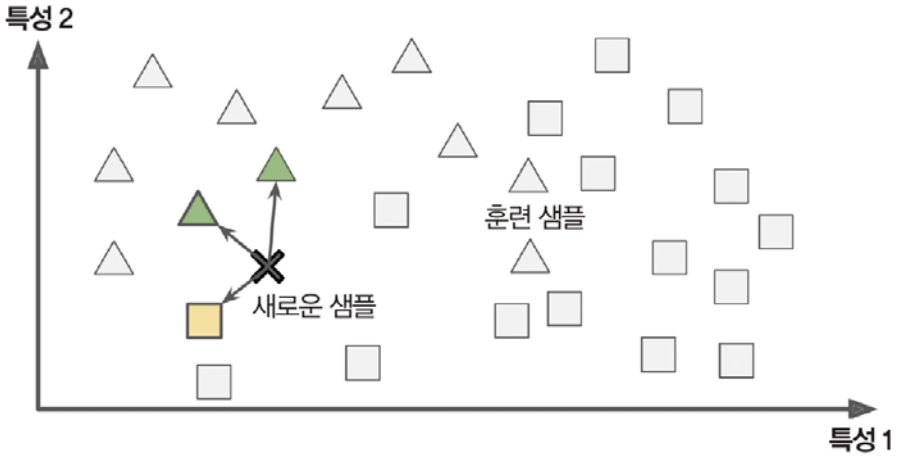


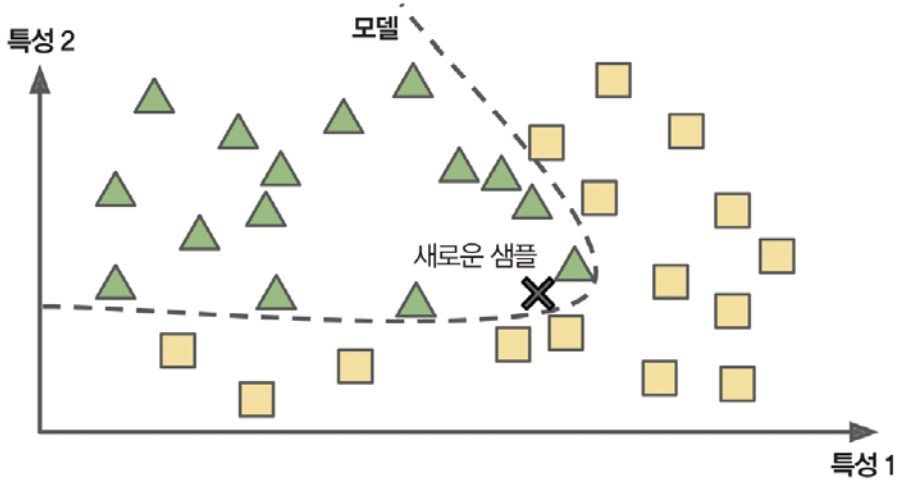
# 머신 러닝

- 온라인 학습
  - 데이터를 순차적으로 한 개 또는 **미니배치**라 부르는 작은 묶음 단위로 주입하여 시스템을 훈련
  - 매 학습 단계가 빠르고 비용이 적게 들어 데이터가 도착하는대로 바로 학습 가능
  - 연속적으로 데이터를 받고 빠른 변화에 스스로 적응해야 하는 시스템에 적합
  - 아주 큰 데이터를 학습하는 시스템에도 온라인 학습 알고리즘 사용 가능(데이터의 일부를 입력으로 학습하는 것을 반복)
  - **학습률**: 변화하는 데이터에 얼마나 빠르게 적응할 것인지
- 배치 학습(오프라인 학습)
  - 시스템이 점진적으로 학습할 수 없음
  - 가용한 데이터를 모두 사용해야 하므로 오프라인으로 수행
  - 시스템을 훈련시키고 제품 시스템에 적용하면 더 이상의 학습은 없음

# 머신 러닝

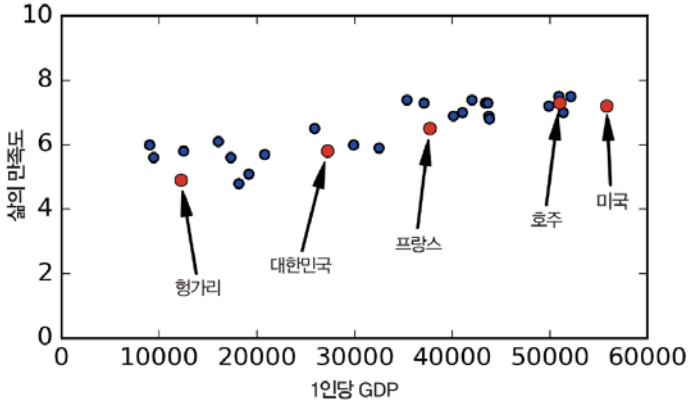
- 사례 기반 학습
  - 스팸 메일과 유사한 메일을 구분하도록 스팸 필터를 프로그래밍할 수 있음
  - 두 메일 사이의 **유사도를 측정**(공통으로 포함된 단어의 수를 세는 것)
  - 시스템이 훈련 샘플을 기억함으로써 학습하는 방식
- 모델 기반 학습
  - 샘플로부터 일반화시키는 다른 방법은 샘플들의 모델을 만들어 예측에 사용하는 것

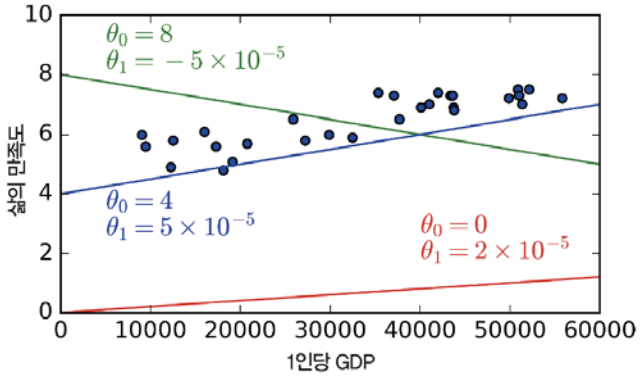


# 머신 러닝

- 모델기반 학습 사례




$$\text{삶의\_만족도} = \theta_0 + \theta_1 \times \text{1인당\_GDP}$$

모델 파라미터

# 머신 러닝

- 머신 러닝 알고리즘

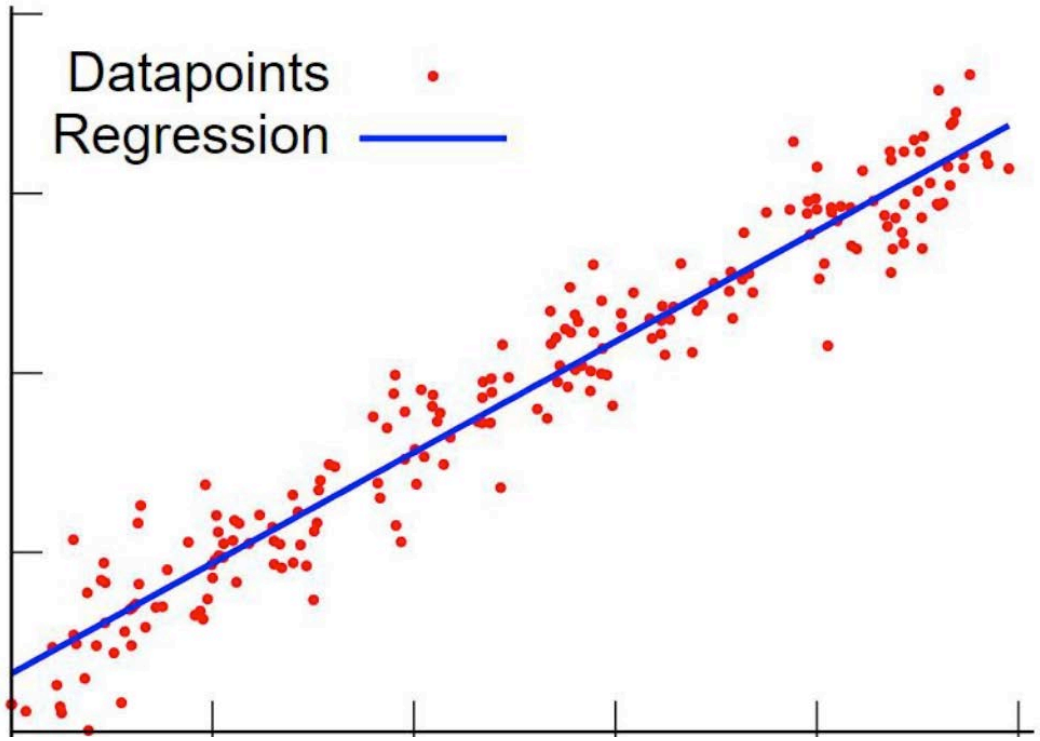
| 항목 | 머신 러닝 유형 | 알고리즘 |
|---|---|---|
| 지도학습 | 분류 | kNN, 베이즈, 결정 트리,<br>랜덤포레스트, 로지스틱회귀, 그라디언트부스팅,<br>신경망 |
| | 회귀 | 선형회귀분석, SVM(서포트 벡터 머신), 신경망 |
| 비지도학습 | 군집화 | k-means, DBASCAN, 계층적군집분석(HCA),<br>이상치 탐지와 특이치 탐지 |
| | 시각화와<br>차원축소 | PCA, 시각화(T-sne), 지역적 선형 임베딩(LLE) |
| | 연관 규칙 | Aprori, Eclat |

# 머신 러닝

- 머신 러닝 동작
  - 머신 러닝은 모델(model)을 사용
    - 스팸 메일을 찾아내는 모델,
    - 누가 게임에서 이길지 예측하는 모델,
    - 내일 날씨를 예측하는 모델
  - 과학에서는 어떤 현상을 설명하는 모델로 수식을 주로 사용
    - 모든 질량을 가진 모든 물체는 서로 끌어당긴다는 만유인력 법칙은 두 물체의 질량에 각각 비례하고, 두 물체의 거리의 자승에 반비례하는 수식으로 표현
    - 머신 러닝, AI 모델은 데이터 기반의 모델을 사용함
- 모델의 가치
  - 와인 품질 = 12.145 + (0.00117x겨울철 강수량)
    + (0.064x 재배철 평균기온) - (0.00386x 수확기 강수량)

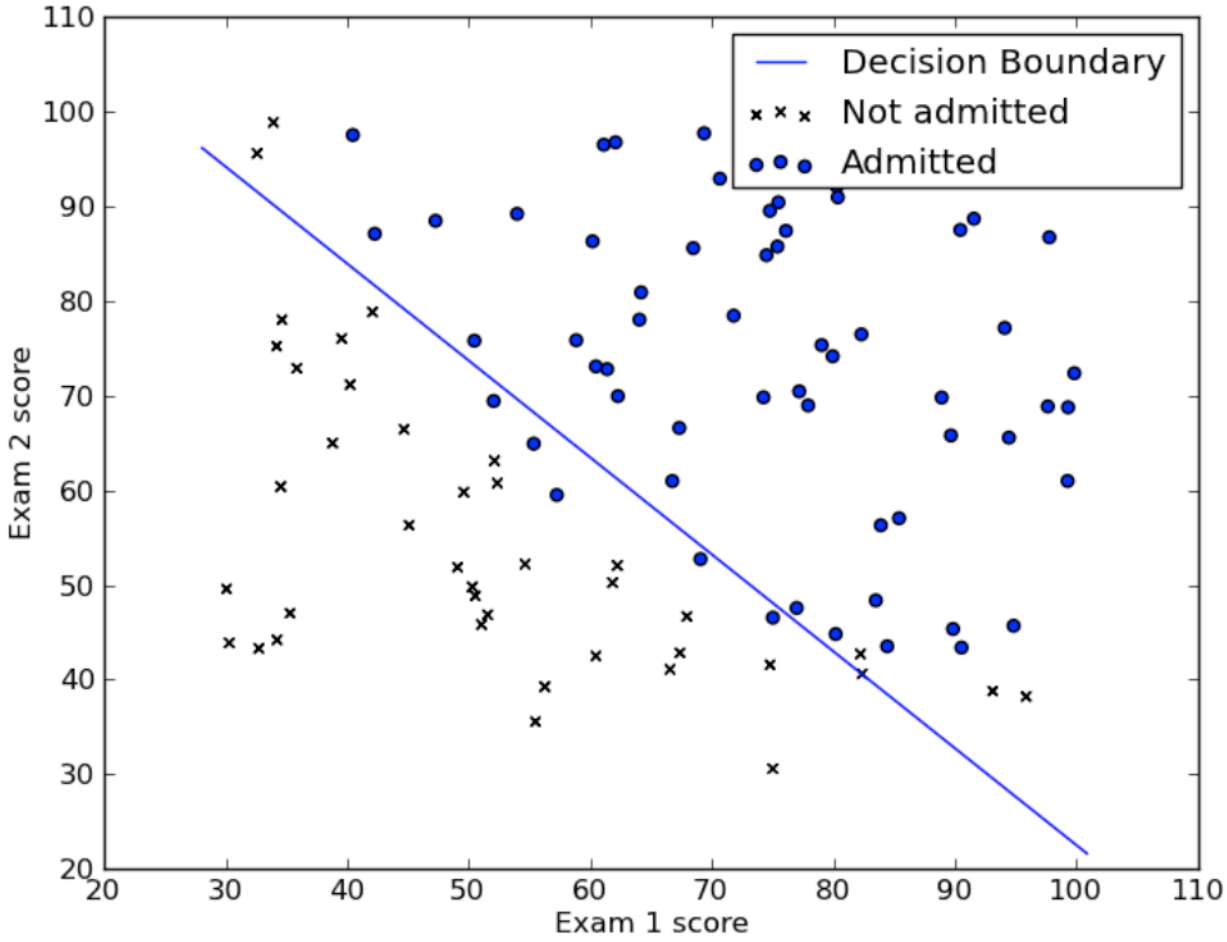
# 머신 러닝

- 선형 회귀(regression)  y = wX + b



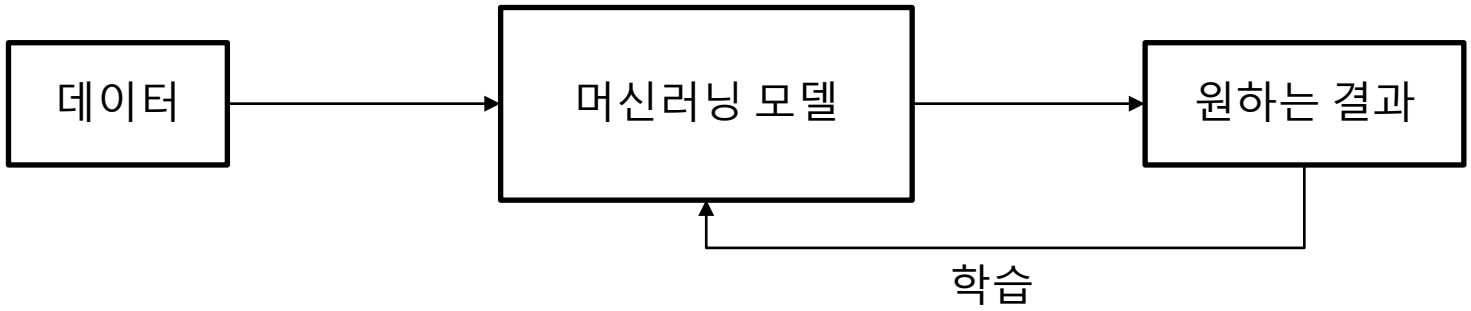
# 머신 러닝

- 선형 분류(classification)  $ay + bx > c$



# 머신 러닝

- 모델의 특징
  - 머신 러닝에서는 데이터에 기반한 모델을 사용 (학습)
  - 현실 세계의 많은 현상은 수식으로 간단히 모델링하기 어렵고 과학적으로 증명할 수는 없음
- 모델 구조와 파라미터
  - 모델 구조: 모델의 동작을 규정하는 방법
  - 모델 파라미터: 모델이 잘 동작하도록 정한 가중치 등 계수
    - 예: 머리카락 길이
    - 모델의 구조는 프로그래머가 선택
    - 적절한 파라미터를 찾는 것은 머신 러닝 프로그램이 학습하여 찾음



# 머신 러닝

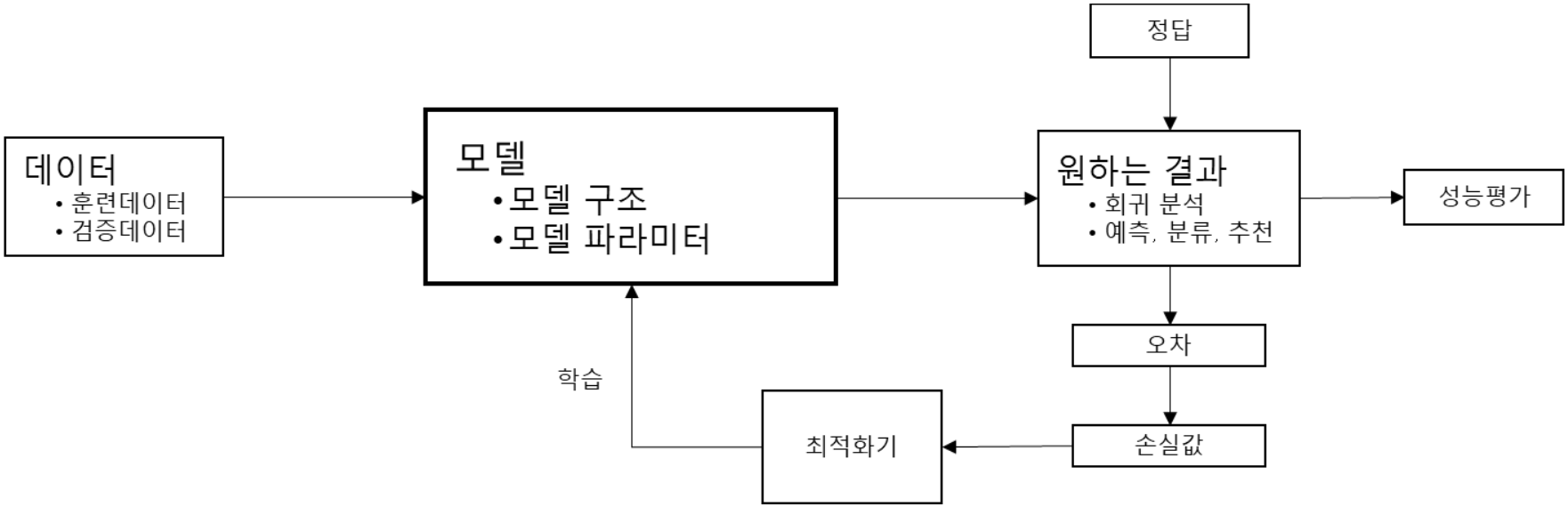
- 손실함수(비용 함수)
  - 모델의 예측 값과 실제 값과의 차이, 즉 오차로부터 손실함수(loss function)을 계산함
  - 이 손실함수를 줄이는 방향으로 모델을 최적화 (학습) 함
  - 회귀분석에서 많이 사용하는 손실함수로는 오차 자승의 합의 평균치(MSE: mean square error)
  - N: 배치 크기

$$MSE = \sum_{k=1}^{N} (y - \hat{y})^2$$

  - 배치 크기 같은 설정 환경 변수를 하이퍼파라미터라고 함
  - 하이퍼파라미터는 사람이 선택하는 변수이며, 기계 학습으로 자동으로 갱신되는 변수는 "파라미터"라고 함

# 머신 러닝

- 오차 손실함수, 최적화, 파라미터



# 머신 러닝

- 분류의 손실 함수
  - 분류에서는 손실함수로 MSE를 사용할 수 없음
  - 대신, 분류에서 정확도(accuracy)를 손실함수로 사용할 수 있음
  - 예를 들어 100명에 대해 남녀 분류를 시도하였으나 96명을 맞추고 4명을 틀렸다면 정확도는 0.96
  - 그러나 정확도를 손실함수로 사용하는데 다음과 같은 문제가 있음
- 카테고리 분포 불균형시 문제
  - 남자가 95명, 여자가 5명이 있는 그룹에서 남자는 1명을 잘 못 분류하고 여자는 3명을 잘 못 분류했다고 하면, 정확도는 여전히 0.96임
  - 손실을 제대로 측정하지 못함
  - 이를 보완하기 위해서 크로스 엔트로피(cross entropy)를 사용함

# 머신 러닝

- 크로스 엔트로피

$$CE = \sum_i p_i \log\left(\frac{1}{p_i'}\right)$$

  - Pi는 어떤 사건이 일어날 실제 확률이고, Pi' 는 예측한 확률이다
  - 남녀가 50명씩 같은 경우

$$CE = -0.5 \times \log\left(\frac{49}{50}\right) - 0.5 \times \log\left(\frac{47}{50}\right) = 0.02687$$

  - 남자가 95명 여자가 5명인 경우

$$CE = -0.95 \times \log\left(\frac{94}{95}\right) - 0.05 \times \log\left(\frac{2}{5}\right) = 0.17609$$

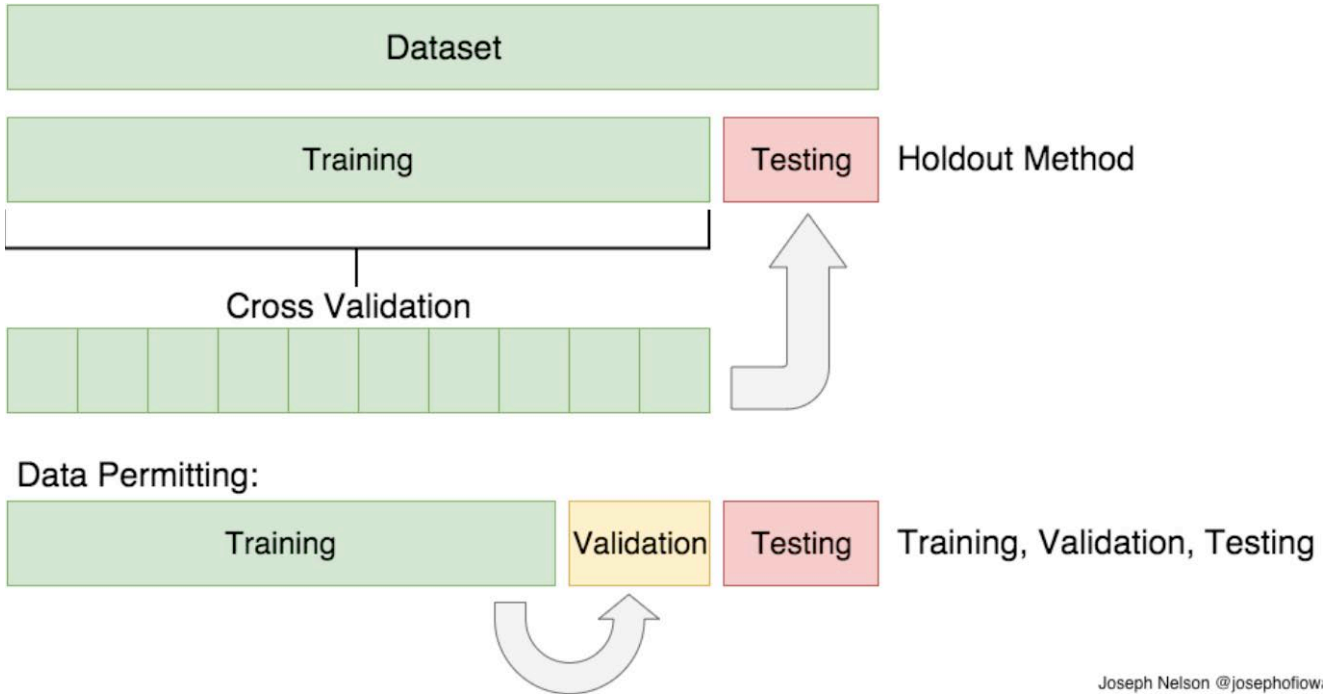
# 머신 러닝

- 훈련과 검증
  - 모델이 데이터를 이용하여 학습하는 과정을 훈련 (training)이라고 함
  - 최적화 알고리즘에 의해서 파라미터(가중치 등)를 계속 갱신하여 모델의 예측 값이 실제 값에 수렴하도록 하는 것
  - 검증(validation) : 훈련된 모델이 잘 동작하는지 확인하는 과정

- 모델 동작이 얼마나 우수한지를 검증할 때는 훈련 데이터로 해서는 안되며 훈련에 사용하지 않은, 새로운 검증 데이터(validation data)를 사용해야 함
- 보통 검증 데이터를 따로 제공하지 않으므로 훈련에 사용할 데이터의 일부를 검증용으로 미리 확보해야 함
- 훈련에 사용하지 않고 남겨 두었다가 모델이 제대로 동작하는지 테스트할 때 사용하는 데이터를 hold-out 데이터라고 함

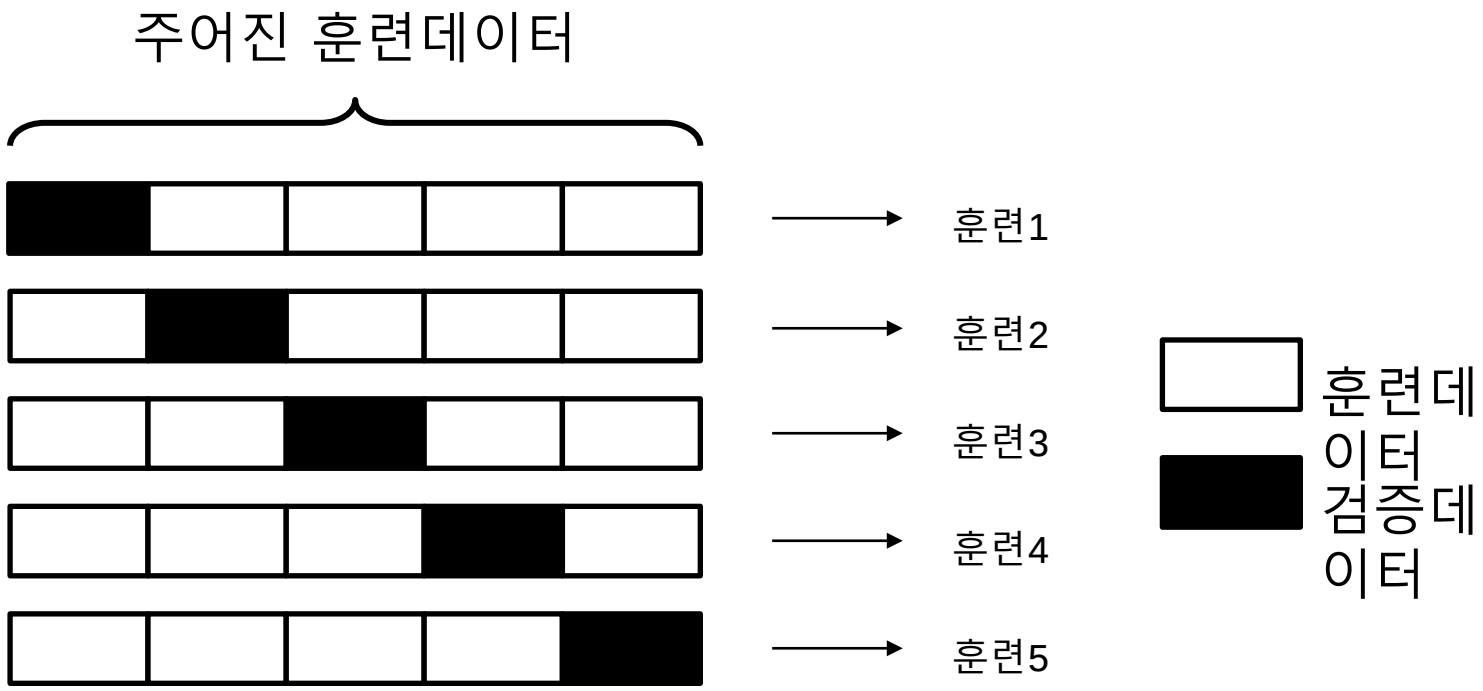
# 머신 러닝

- 훈련과 검증
  - 훈련 데이터 : 모델 파라미터를 훈련하는데 사용
  - 검증 데이터 : 과대적합이나 과소적합을 검사하고 최적 모델 구조(하이퍼파라미터 등)를 찾는데 사용
  - 테스트 데이터 : 모델의 성능을 최종적으로 테스트 하는데 사용



# 머신 러닝

- K-fold 교차 검증

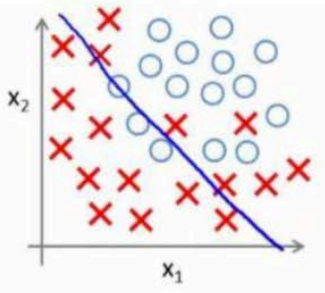


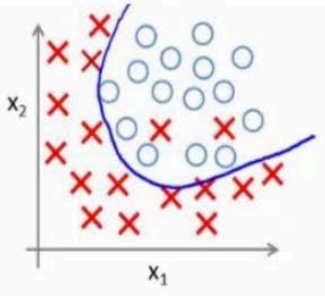
# 머신 러닝

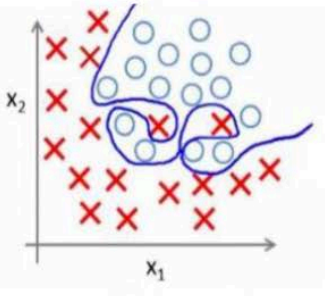
- 데이터의 대표성
  - 훈련 데이터가 미래에 나타날 가능성이 있는 모든 데이터의 특징을 반영하도록 구성해야 함
    - 예: 지리적, 인종적, 나이별, 소득별, 성별 등 균일성 유지
  - 훈련, 검증, 테스트 샘플 데이터가 전체 데이터의 특징을 계속 유지할 수 있어야 함

# 머신 러닝

- 과대적합(over fitting)
  - 모델이 훈련 데이터에 대해서만 잘 동작하도록 훈련되어 새로운 데이터에 대해서는 오히려 잘 동작하지 못하는 것
  - 과대적합된 모델은 훈련 데이터에 대해서는 매우 우수한 성능을 보이지만 일반화가 떨어짐
  - 머신러닝에서는 과대적합을 피해서 일반적으로 잘 동작하게 모델을 만드는 것이 매우 중요함
    - 이를 모델의 일반화(generalization)라고 함




- 과소 적합(under fitting)
  - 모델이 너무 간단하여 성능이 미흡한 경우
  - 과소적합을 피하려면 좀 더 상세한 모델 구조를 사용해야 함
  - 머신러닝에서는 과대적합과 과소적합을 모두 피해야 하며 최적의 예측을 수행하는 모델을 만드는 것이 중요함

# 머신 러닝

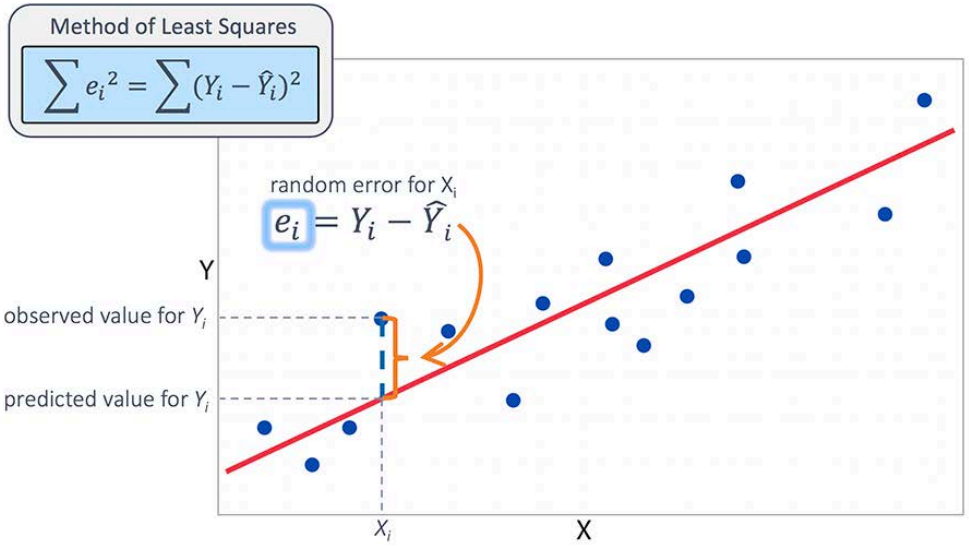
- 모델의 성능
  - 모델의 성능을 평가하는 척도 필요
  - 분류에서는 정확도(accuracy)를 성능 척도로 주로 사용
    - (참고) 분류에서 손실함수로 크로스 엔트로피를 주로 사용
  - 손실함수와 성능 지표의 차이점
    - 손실함수를 정하는 목적은 모델을 훈련시킬 때의 기준으로 삼기 위해서임
    - 모델은 손실함수를 최소화 하는 방향으로 학습
    - 모델의 성능은 이렇게 만든 모델이 궁극적으로 얼마나 잘 동작하는지를 평가하는 척도임

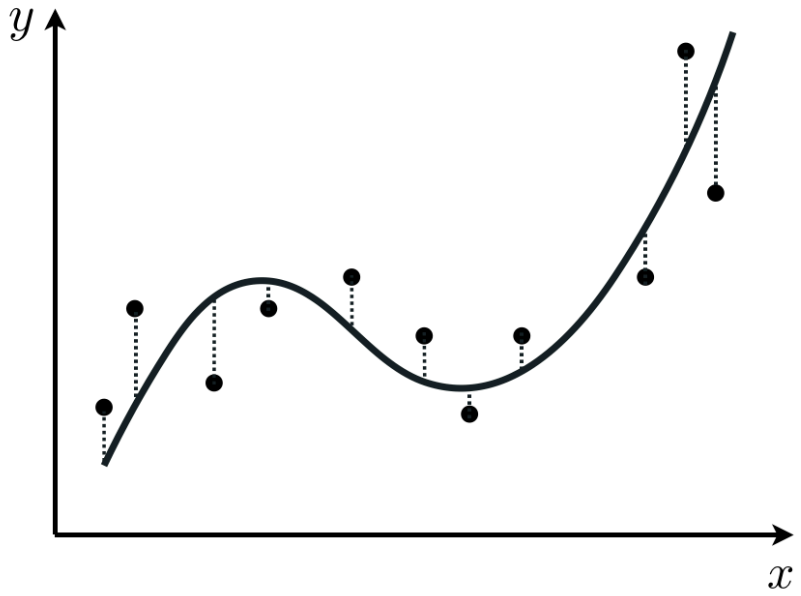
| | 손실함수 | 성능 지표 |
|---|---|---|
| 정의 | 손실함수를 줄이는 방향으로 모델이 학습을 함 | 성능을 높이는 것이 머신러닝을 사용하는 목적임 |
| 회귀 모델 | MSE (오차 자승의 평균) | R2 |
| 분류 모델 | 크로스 엔트로피 | 정확도, 정밀도, 재현률, F1점수 |

# 회귀 분석

- 회귀 분석
  - 수치형 종속변수와 수치형 독립변수사이의 영향 또는 인과관계를 알 수 있는 분석
  - 학습 데이터 x로 부터 y를 예측하는 함수 f(x)를 찾는 과정으로 x와 y는 모두 연속적인 수치 값
  - 도출된 회귀식에서 직선의 기울기와 상수를 알 수 있는데, 이를 통해 독립변수의 변화에 따른 종속변수의 변화를 알 수 있는 것

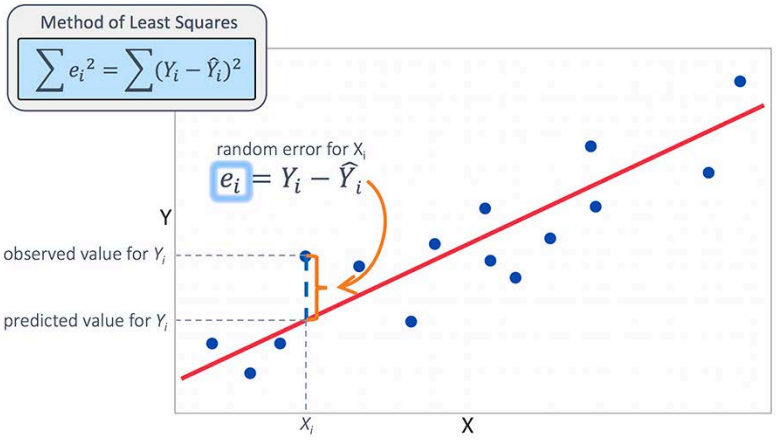
$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i + \cdots + \beta_k X_i + \epsilon_i \, , (i = 1, \cdots, n)$$



# 회귀 분석

- 회귀 문제와 손실함수
  - MAE(Mean absolute Error): 원본 값과 예측 값에 대한 절대 오류의 평균
  - MSE(Mean Squared Error): 원본 값과 예측 값에 대한 오류 제곱의 평균
  - RMSE(Root MSE): MSE의 제곱근
  - R-squared: 원본 값과 예측 값을 비교하여 회귀모델이 얼마나 잘 원본 값을 나타내는지 [0,1]




$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where,

$\hat{y}$ – *predicted value of* $y$

$\bar{y}$ – *mean value of* $y$

# 회귀 분석

- 단순 회귀 분석
  - 하나의 수치형 설명변수가 하나의 수치형 종속변수에 어떤 인과관계 또는 영향을 미치는지에 대한 분석을 말함
  - 많은 변수는 고려하지 않고 오직 하나의 종속변수(Y)와 하나의 독립변수(X)에 의해서만 시행
  - Ex) 쇼핑몰의 입점 매장 수가 고객의 방문빈도에 어떤 영향을 미치는지를 확인하려 하는 상황에 우리는 단순회귀분석을 사용

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i, (i = 1, \cdots, n)$$

$$\hat{Y} = b_0 + b_1 X$$

| | 총매출액 | 방문빈도 | 1회평균매출액 | 쿠폰사용횟수 | 거래기간 |
|---|---|---|---|---|---|
| 0 | 12717240 | 109 | 116672 | 4 | 1093 |
| 1 | 12802210 | 22 | 581919 | 20 | 1002 |
| 2 | 12815010 | 27 | 474630 | 11 | 1066 |
| 3 | 13038990 | 24 | 543291 | 5 | 1069 |
| 4 | 13072260 | 37 | 353304 | 9 | 1077 |

# 회귀 분석

- 단순 회귀 분석
  - P쇼핑몰에서 최근 인기있는 S브랜드매장을 입점시킨 결과, 고객들의 전반적인 방문빈도가 늘어났다고 함
  - 늘어난 방문빈도가 실제 총 매출액에 영향을 미치는지 알아보고자

```
model = smf.ols(formula = '총매출액 ~ 방문빈도', data = data)
result = model.fit()
result.summary()
```

$$\hat{Y}_i = 13490000 + 144800 \times \text{'방문빈도'}$$

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | 총매출액 | **R-squared:** | 0.191 |
| **Model:** | OLS | **Adj. R-squared:** | 0.184 |
| **Method:** | Least Squares | **F-statistic:** | 26.64 |
| **Date:** | Mon, 25 Feb 2019 | **Prob (F-statistic):** | 1.06e-06 |
| **Time:** | 15:56:14 | **Log-Likelihood:** | -2015.2 |
| **No. Observations:** | 115 | **AIC:** | 4034. |
| **Df Residuals:** | 113 | **BIC:** | 4040. |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 1.349e+07 | 1.89e+06 | 7.129 | 0.000 | 9.74e+06 | 1.72e+07 |
| 방문빈도 | 1.448e+05 | 2.81e+04 | 5.161 | 0.000 | 8.92e+04 | 2e+05 |

# 회귀 분석

- 다중 회귀 분석
  - 하나의 수치형 종속변수와 2개이상의 수치형 독립변수 사이의 영향 또는 인과관계를 설명하는 분석을 말함
  - 예를 들어, 아빠의 키, 엄마의 키, 할머니의 키(독립변수)가 자녀의 키(종속변수)에 어떤 관계를 갖는지 확인하려 할 때 우리는 다중회귀분석을 사용함

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_i X_i + \epsilon_i$$

$$\hat{Y} = b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_i X_i$$

  - 변수선택법: 다중회귀분석을 할 때는 여러 개의 독립변수를 선택하므로 최적화된 회귀모형을 만들기 위해서는 어떤 변수들을 독립변수에 넣을 것인지를 잘 판단해야 함

| 독립변수 선택 방법 | |
|---|---|
| 입력 | 사용하고자하는 독립변수를 모두 입력하는 방식 |
| 전진 | 모형적합에 가장 큰 영향을 미치는 독립변수를 순서대로 추가하는 방식 |
| 후진 | 모형적합에 가장 약하게 영향을 미치는 독립변수를 순서대로 제거하는 방식 |
| 단계 선택 | 전진선택과 후진선택을 결합한 방식이다. 전진선택을 사용하여 독립변수를 추가한 뒤, 다중공선성을 판단하여 후진방식으로 독립변수를 제거하는 방식이다. |

# 회귀 분석

- 다중 회귀 분석

```
model = smf.ols(formula = '방문빈도 ~ 거래기간 + 총매출액 + 쿠폰사용횟수', data = data)
result = model.fit()
result.summary()
```

$$\hat{Y}_i = -55.174 + 0.0786 * 거래기간 + 0.000001 * 총매출액$$

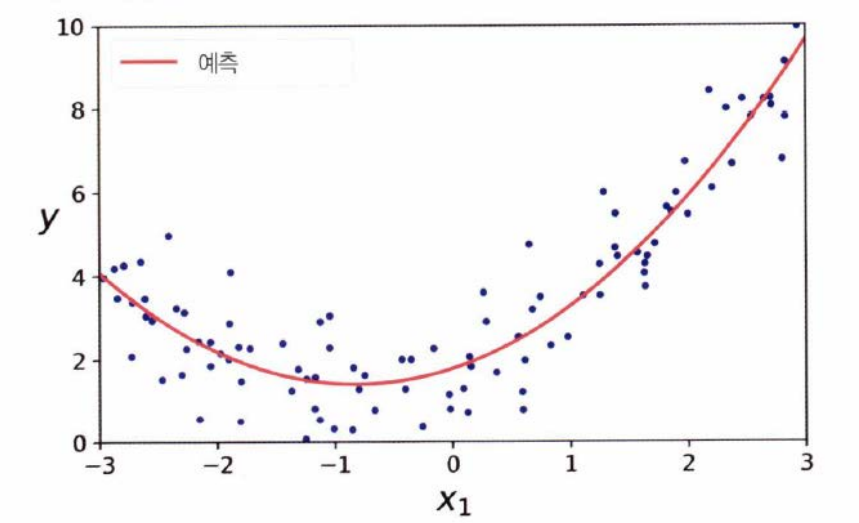
OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | 방문빈도 | **R-squared:** | 0.245 |
| **Model:** | OLS | **Adj. R-squared:** | 0.225 |
| **Method:** | Least Squares | **F-statistic:** | 12.02 |
| **Date:** | Mon, 25 Feb 2019 | **Prob (F-statistic):** | 7.14e-07 |
| **Time:** | 15:56:14 | **Log-Likelihood:** | -549.34 |
| **No. Observations:** | 115 | **AIC:** | 1107. |
| **Df Residuals:** | 111 | **BIC:** | 1118. |
| **Df Model:** | 3 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | -55.1744 | 31.260 | -1.765 | 0.080 | -117.117 | 6.768 |
| 거래기간 | 0.0786 | 0.029 | 2.665 | 0.009 | 0.020 | 0.137 |
| 총매출액 | 1.215e-06 | 2.51e-07 | 4.834 | 0.000 | 7.17e-07 | 1.71e-06 |
| 쿠폰사용횟수 | 0.2959 | 0.312 | 0.948 | 0.345 | -0.323 | 0.915 |

# 회귀 분석

- 다항 회귀 분석
  - 한 특성과 예측값의 관계가 선형이 아닌 2차, 3차 이상의 관계를 갖는 회귀 방법

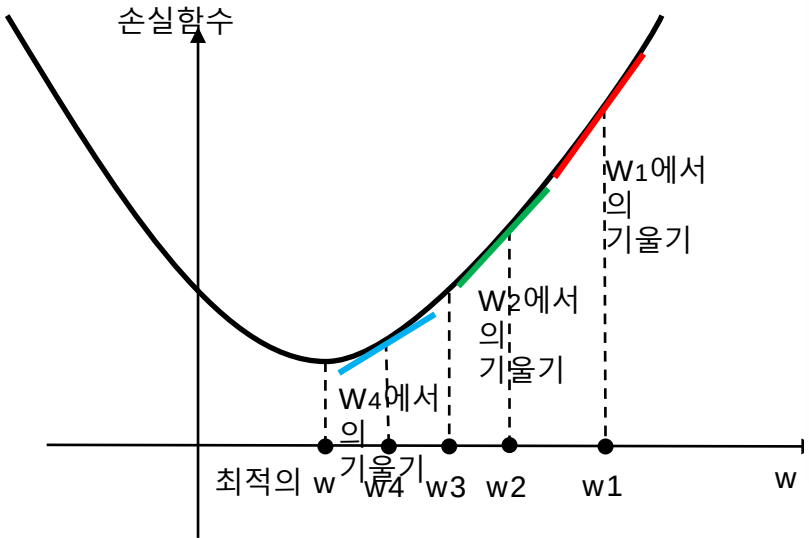


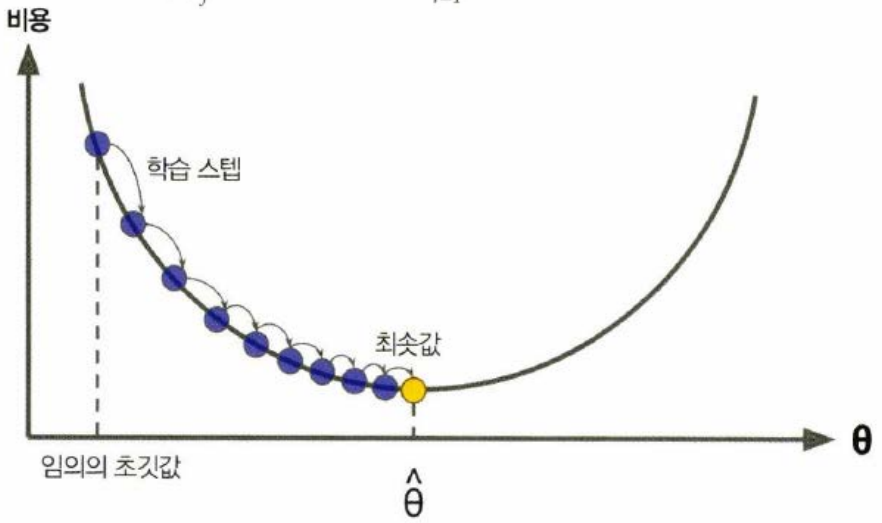
# 회귀 분석

- 최적화(경사 하강법)
  - 가장 일반적인 최적화 알고리즘: (Gradient Descent)
  - 손실함수를 계수에 관한 그래프로 그렸을 때 최소값으로 빨리 도달하기 위해서는 현재 위치에서의 기울기(미분 값)에 비례하여 반대방향으로 이동하는 방식
- 경사 하강법 특징
  - 경사 하강법을 적용하려면 특성 변수들을 모두 동일한 방식으로 스케일링해야 한다.
  - 특성 값마다 크기의 편차가 크면 특정 변수에 너무 종속되어 동작할 수 있고 이로 인해 수렴속도가 직선이 되지 않고 오래 걸릴 수가 있다.

$$W_i = W_{i-1} - \eta Grad(i)$$

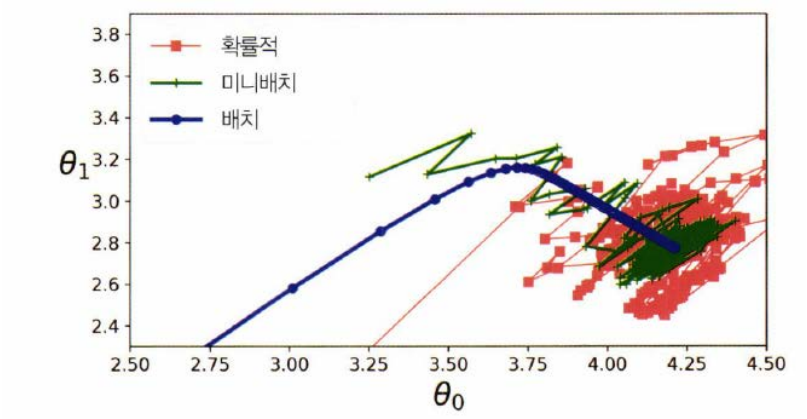
$$\frac{\partial}{\partial \theta_j} \mathrm{MSE}(\boldsymbol{\theta}) = \frac{2}{m} \sum_{i=1}^{m} \left( \boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)} \right) x_j^{(i)}$$



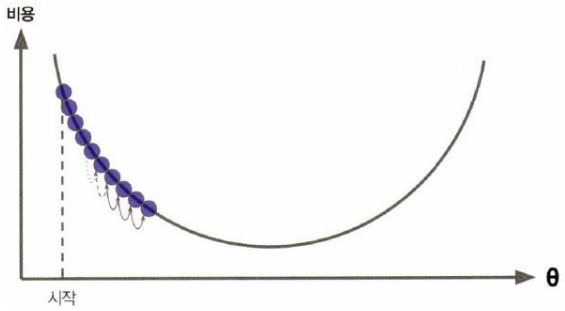
# 회귀 분석

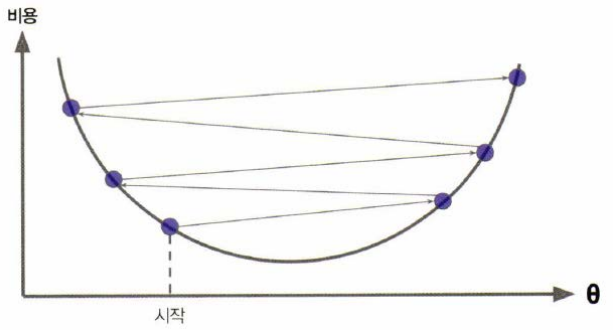
- 경사 하강법 종류
  - 배치(Batch) GD
    - 일반적으로 배치 GD방식을 많이 사용하는데, 적절한 크기의 배치단위로 입력 신호를 나누어 경사 하강법을 적용하는 방식임
  - SGD (확률적 경사 하강법)
    - 한 번에 한 샘플씩 랜덤하게 골라서 훈련에 사용하는 방법이다.
    - 즉 샘플을 하나만 보고 계수를 조정함
    - 계산량이 적어 동작속도가 빠르고, 랜덤한 방향으로 학습을 하므로 전역 최소치를 가능성이 높아짐
    - 매 샘플이 너무 랜덤하여 방향성을 잃고 수렴하는데 시간이 오래 걸릴 가능성도 있음

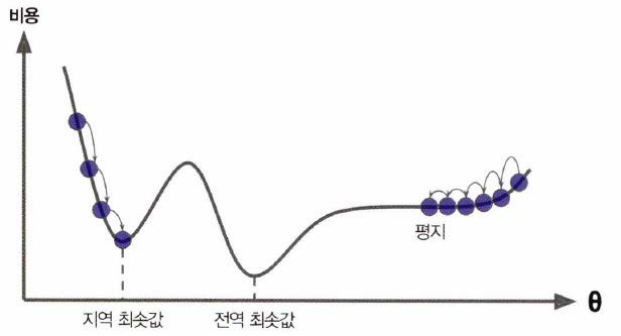


# 회귀 분석

- 학습률
  - 계수를 업데이트 하는 속도를 조정하는 변수
  - 학습률이 너무 작으면 수렴하는데 시간이 오래 걸리지만 최저점에 도달했을 때 흔들림 없이 안정적인 값을 얻게 되고,
  - 학습률을 너무 크게 정하면 학습하는 속도는 빠르나 자칫하면 최저점으로 수렴하지 못하고 발산하거나 수렴하더라도 흔들리는 오차가 남아있을 수 있음
- 학습 스케줄(learning schedule) 기법
  - 초기에는 학습률을 크게 정하고 (학습률을 빠르게 하고) 오차가 줄어들면 학습률을 줄여서 안정상태(steady state)의 오차를 줄이는 방법

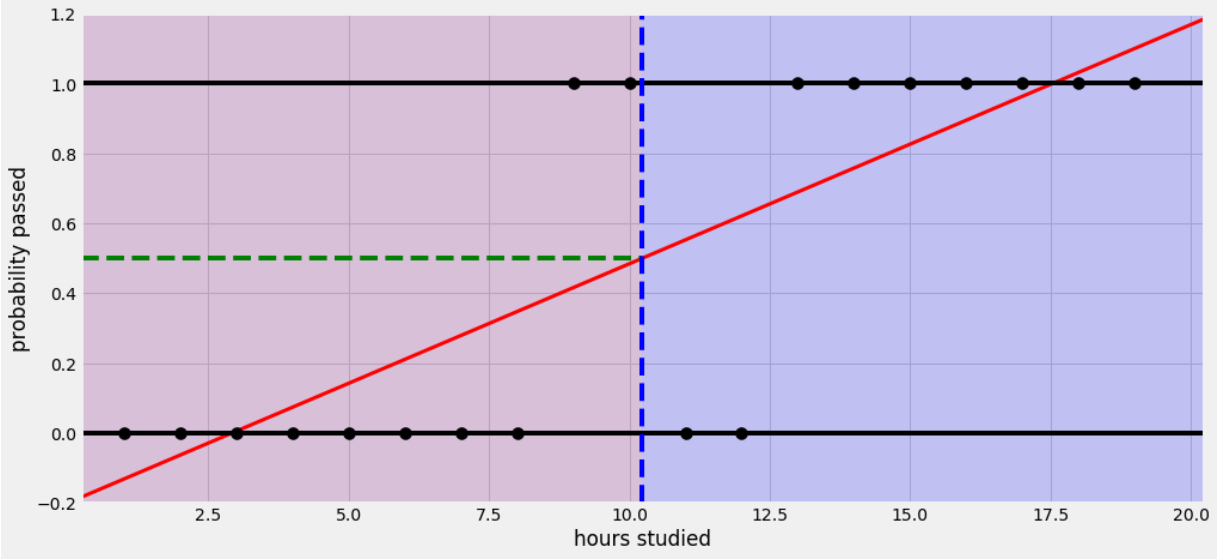


# 회귀 분석

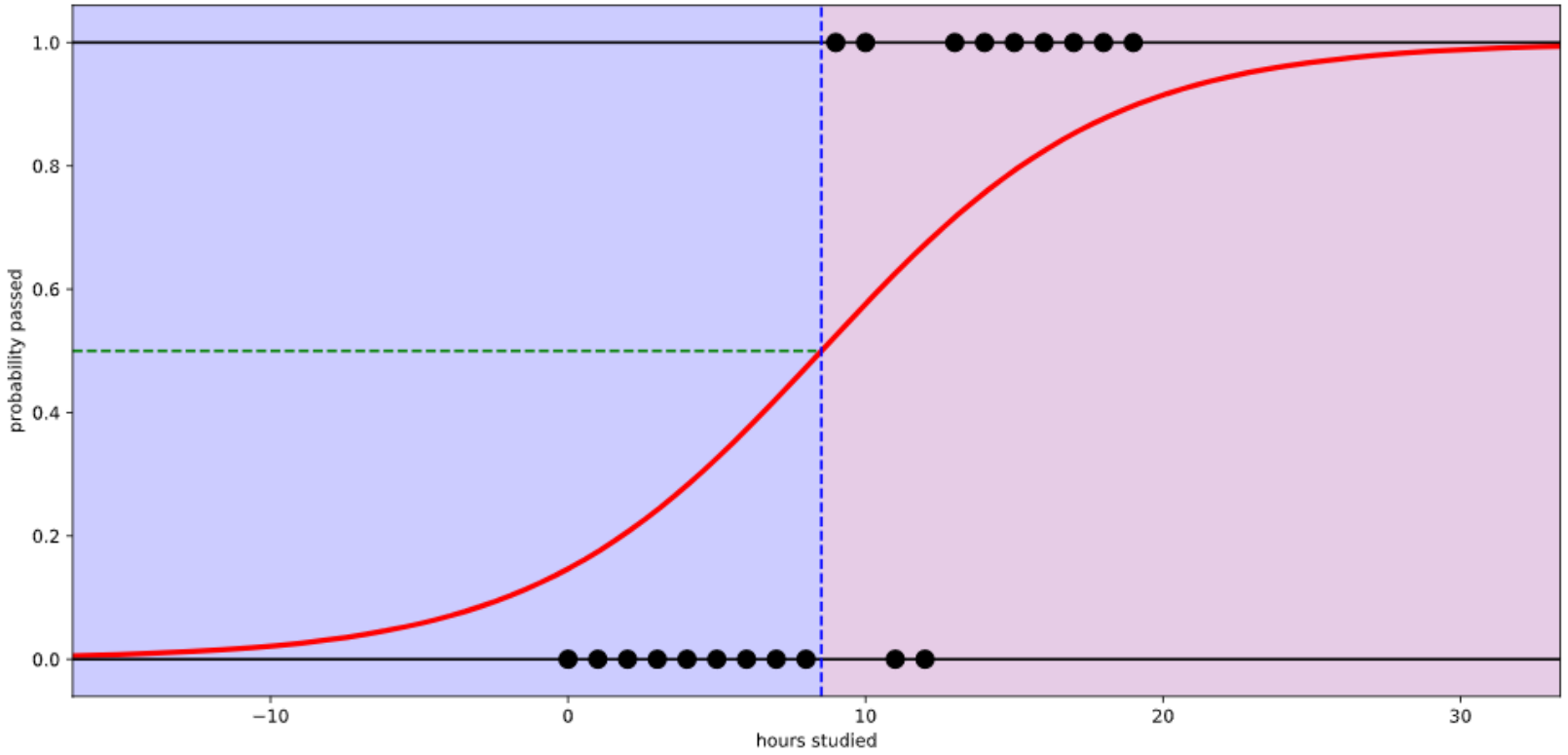
- 로지스틱 회귀
  - 임의의 범위를 갖는 값으로부터 0과 1사이의 값을 예측하거나 이진 분류에 사용하는 알고리즘임
  - 로지스틱 회귀분석은 보통 독립 변수와 종속 변수의 관계를 S형 커브로 매핑함(선형 회귀분석 사용이  불가한 경우)
  - 신용도 판단, 연간 구매량 기준 우수 고객 여부 판단, 평가 지표 기준 합격 여부 판단, 건강 지표에 따른 건강 여부, 팀의 승리/패배 여부 예측 등 여러 경우에 사용함
- 공부시간과 합격 여부




https://hleecaster.com/ml-logistic-regression-concept/

# 회귀 분석

- 로지스틱 회귀
  - 데이터가 특정 범주에 속할 확률을 예측하는 단계
  - 모든 속성(feature)들의 계수(coefficient)와 절편(intercept)을 0으로 초기화
  - 각 속성들의 값(value)에 계수(coefficient)를 곱해서 log-odds를 구함
  - Log-odds를 sigmoid 함수에 넣어서 [0,1] 범위의 확률을 구함



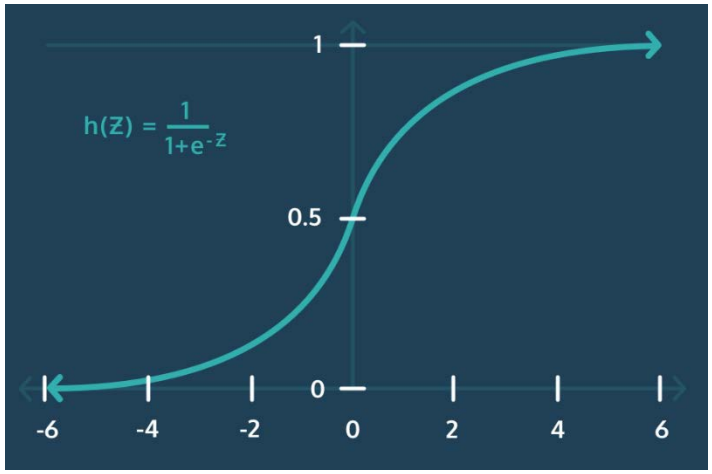
# 회귀 분석

- 로지스틱 회귀
  - odds: 사건이 발생할 확률을 발생하지 하지 않을 확률로 나눈 값

$$Odds = \frac{P(event\ occurring)}{P(event\ not\ occurring)}$$

$$Odds\ of\ passing = \frac{0.7}{0.3} = 2.\overline{33} \qquad Log\ odds\ of\ passing = log(2.\overline{33}) = 0.847$$

$$z = b_0 + b_1 x_1 + \cdots + b_n x_n$$

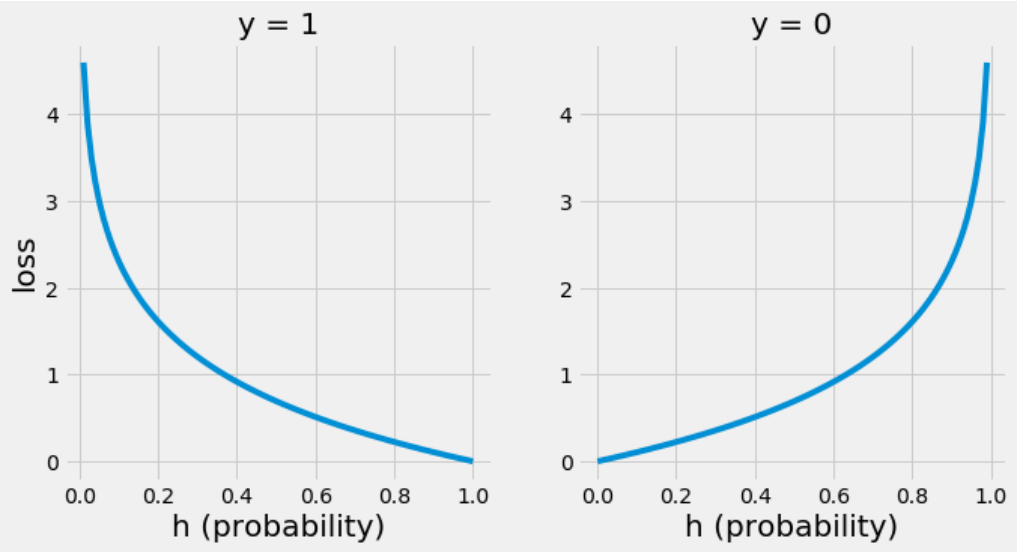



# 회귀 분석

- 로지스틱 회귀
  - 비용 함수(손실 함수): 로지스틱 회귀가 확률을 제대로 예측해주는지, 즉 구해놓은 속성들의 계수(coefficients)와 절편(intercept)이 적절한지 확인하기 위해 손실(Loss)을 고려
  - 모델의 "적합성"을 평가하기 위해 각 데이터 샘플의 손실(모델 예측이 얼마나 잘못되었는지)을 계산한 다음 그것들의 평균화 해야 함

$$-\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h(z^{(i)})) + (1-y^{(i)})log(1-h(z^{(i)}))]$$

$$loss_{y=1} = -log(h(z^{(i)})) \qquad loss_{y=0} = -log(1-h(z^{(i)}))$$



# 회귀 분석

- 다항 로지스틱 회귀(소프트맥스 회귀)
  - 앞에서는 이진 분류, 즉 합격/불합격 등 두 개의 레이블을 가진 경우에 로지스틱 회귀를 사용하는 예를 소개했음
  - 그런데 2개가 아니라 3개 이상의 클래스 중에 하나를 예측해야 하는 경우는 다항 로지스틱 회귀(multinomial logistic regression)를 이용함
  - 소프트맥스 (softmax) 함수를 사용함
    - k : 범주의 수
    - s(x) : 샘플 x에 대한 각 범주의 점수를 담고 있는 벡터
    - σ(s(x))k : 이 샘플이 범주 k에 속할 확률

$$s_k(\mathbf{x}) = (\boldsymbol{\theta}^{(k)})^T \mathbf{x} \qquad \hat{p}_k = \sigma(\mathbf{s}(\mathbf{x}))_k = \frac{\exp\left(s_k(\mathbf{x})\right)}{\sum_{j=1}^{K} \exp\left(s_j(\mathbf{x})\right)}$$

  - Argmax
    - 이 연산은 함수를 최대화하는 변수의 값을 반환한다(numpy에도 비슷한 함수가 있는데 array에서 최댓값을 가지는 원소의 index를 반환)

$$\hat{y} = \underset{k}{\operatorname{argmax}}\, \sigma(\mathbf{s}(\mathbf{x}))_k = \underset{k}{\operatorname{argmax}}\, s_k(\mathbf{x}) = \underset{k}{\operatorname{argmax}}\, \left((\boldsymbol{\theta}^{(k)})^T \mathbf{x}\right)$$

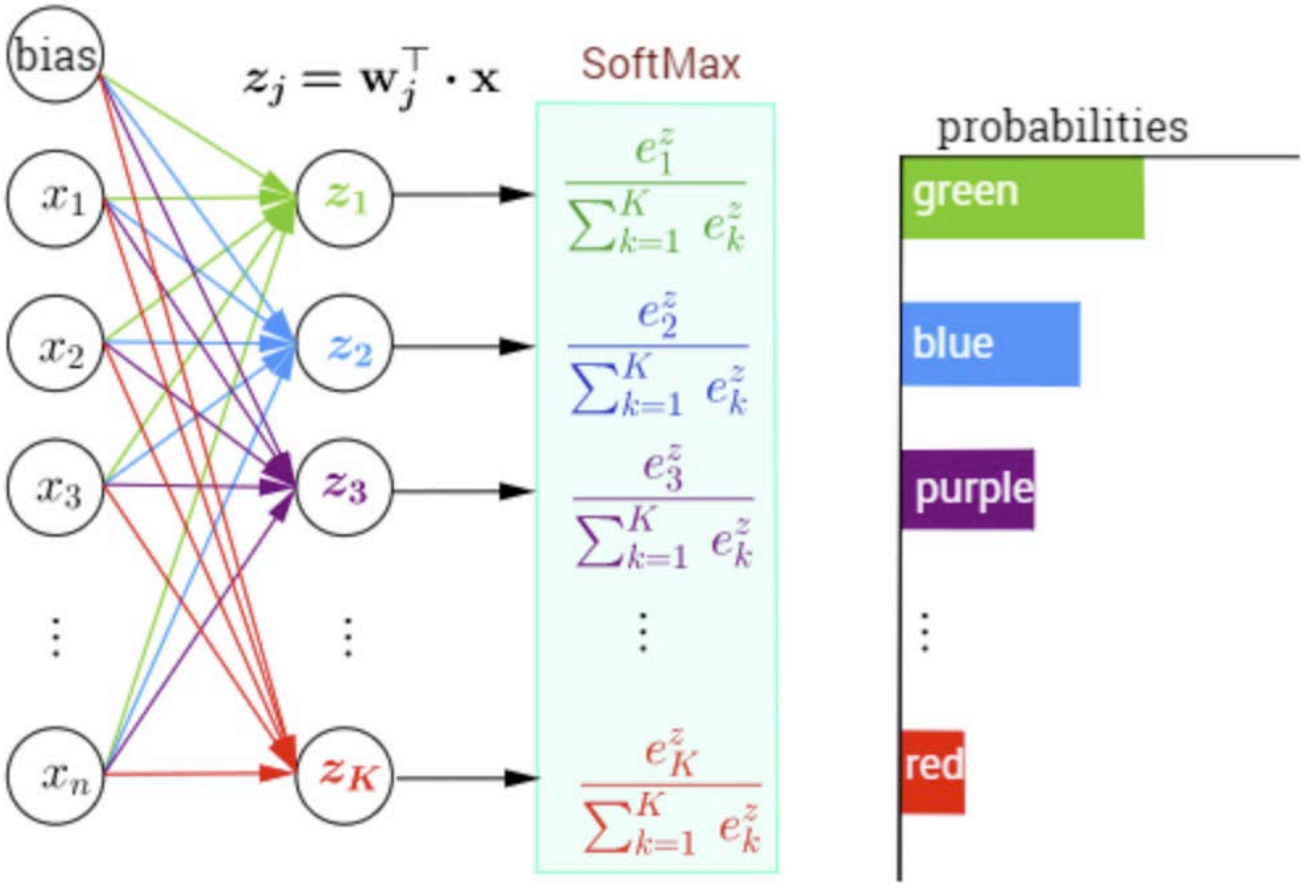
# 회귀 분석

- 다항 로지스틱 회귀(소프트맥스 회귀)
  - 비용 함수 : 크로스 엔트로피

$$J(\mathbf{\Theta}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)$$

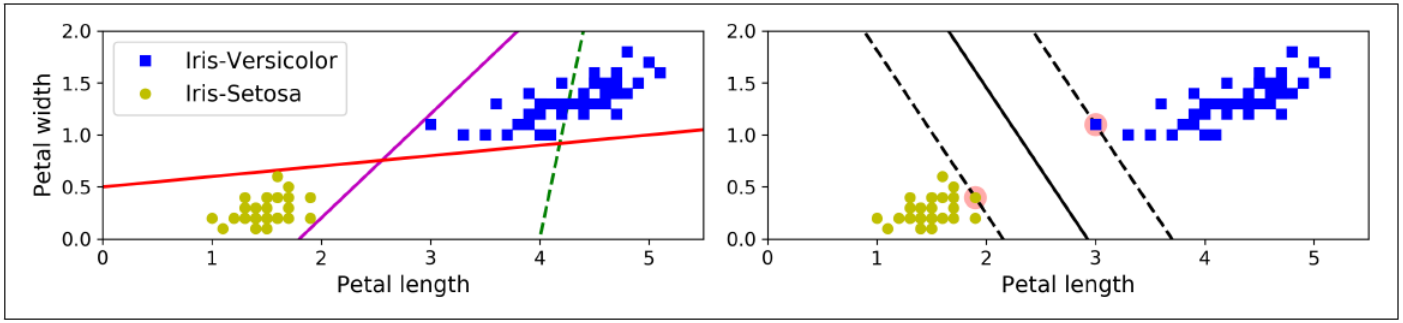
# 회귀 분석

- 소프트맥스


bias
$x_1$
$x_2$
$x_3$
$x_n$
$z_j = \mathbf{w}_j^\top \cdot \mathbf{x}$
$z_1$
$z_2$
$z_3$
$z_K$
SoftMax
$\frac{e_1^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_2^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_3^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_K^z}{\sum_{k=1}^K e_k^z}$
probabilities
green
blue
purple
red

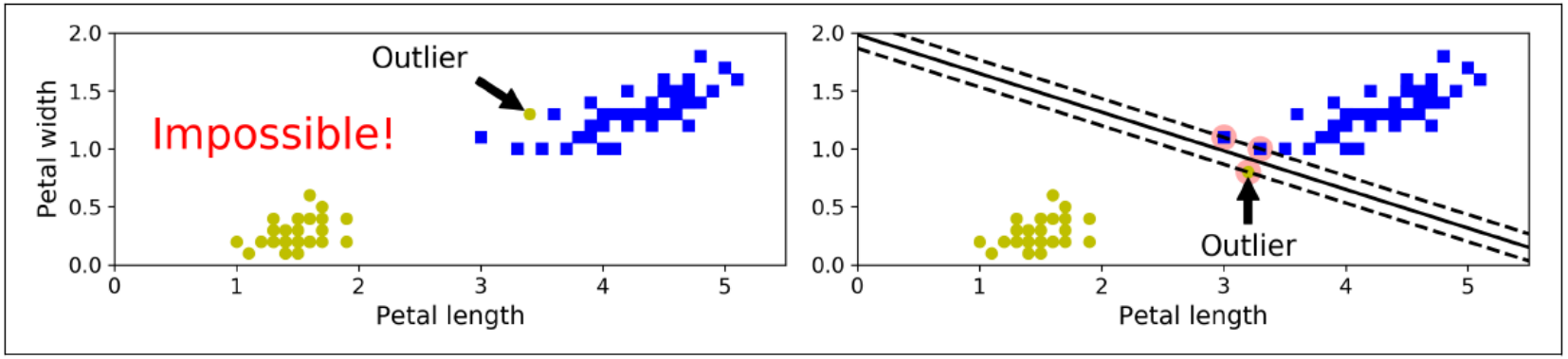
# 회귀 분석

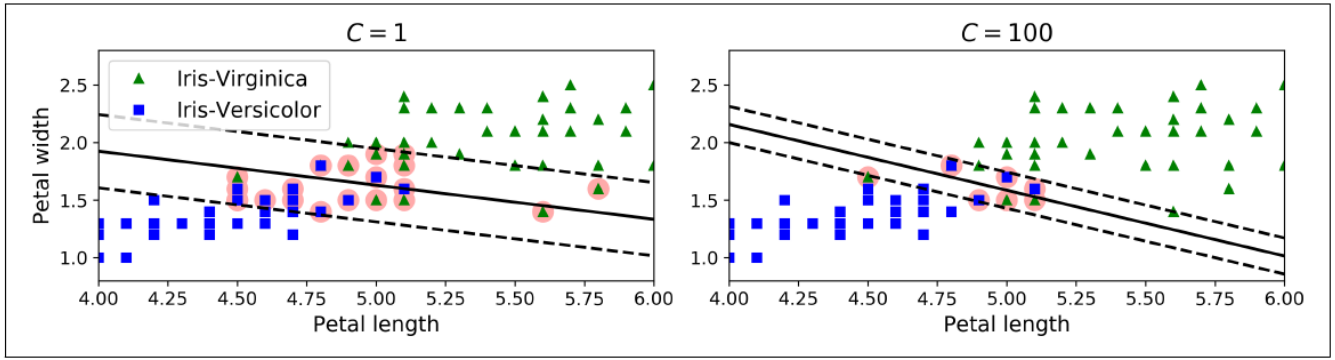
- 서포트 벡터 머신(SVM)
  - SVM은 비선형, 선형 분류, 회귀, 이상치 탐색을 하는데 사용할 수 있는 강력한 ML 모델 중 하나
  - SVM은 특히 복잡한 문제에 잘 맞으며 작거나 중간 크기의 데이터 셋에 적합
- 선형 SVM(라지 마진 분류)



# 회귀 분석

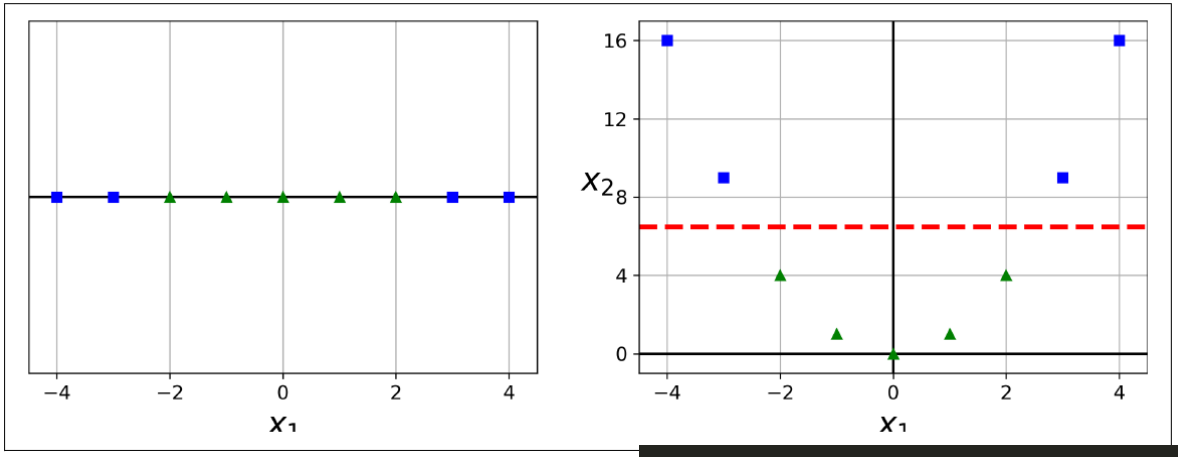
- 서포트 벡터 머신(SVM)
  - 소프트 마진 분류
    - 모든 샘플이 마진 바깥쪽에 올바르게 분류되어 있는 경우를 **하드마진 분류**
      - **데이터가 선형적으로 구분되어야 제대로 동작**
      - **이상치에 민감**
    - 도로의 폭을 가능한 넓게 유지하는 것과 마진 오류 사이의 적절한 균형을 찾는 **소프트 마진 분류**



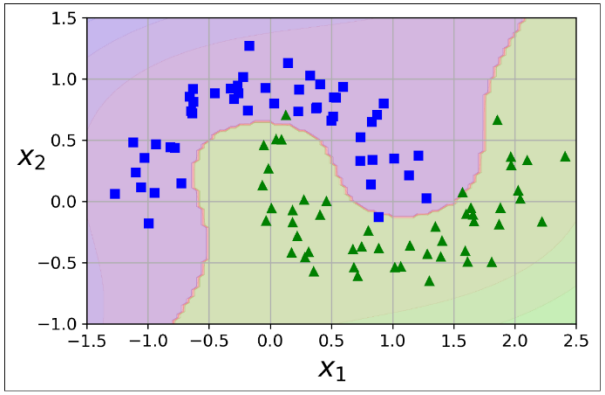



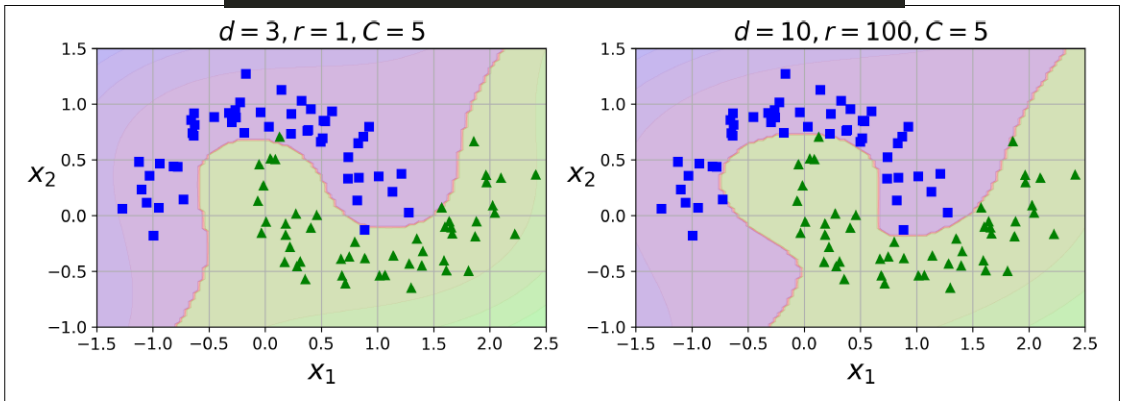
# 회귀 분석

- 비선형 SVM 분류
  - 많은 경우의 데이터는 직선으로 분류되지 않음
  - 특징점들을 추가해서 직선으로 분류 가능하게 만들면 됨 $x_2 = {x_1}^2$



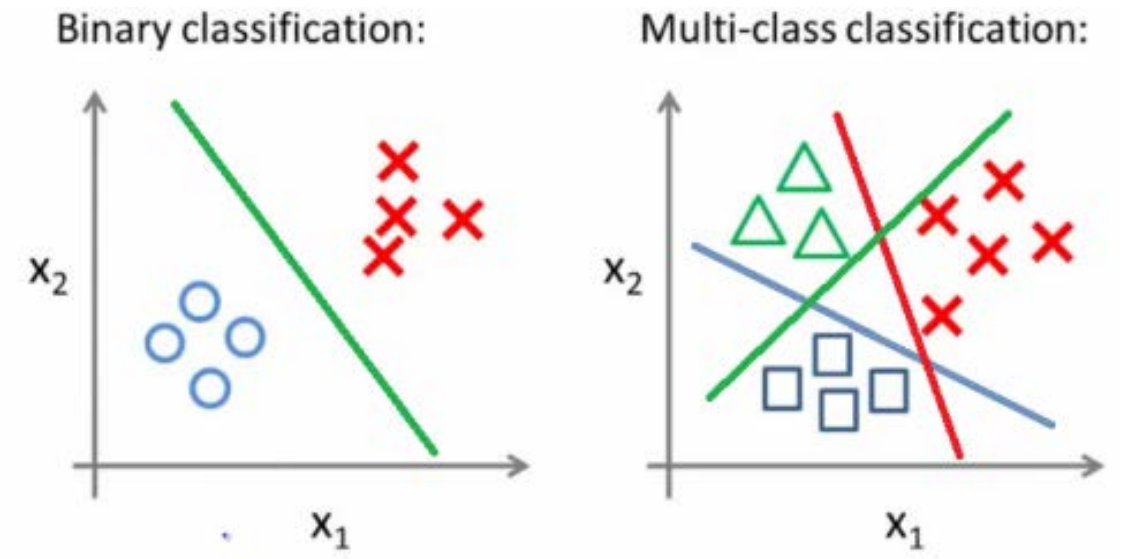

```
("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
```



# 분류

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

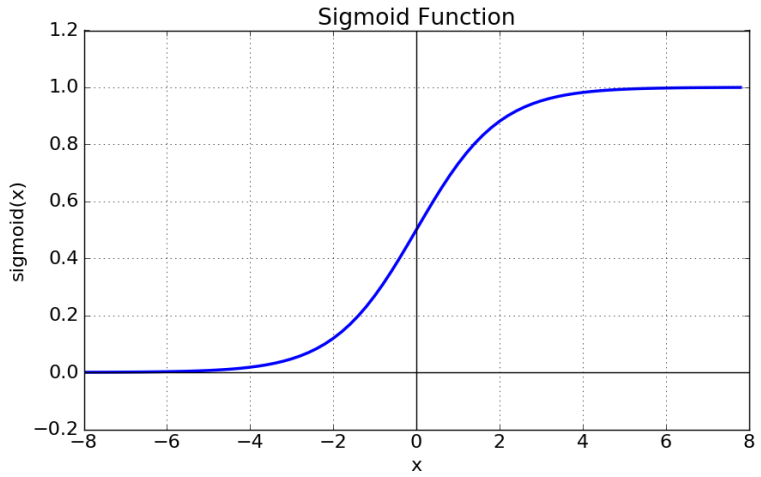


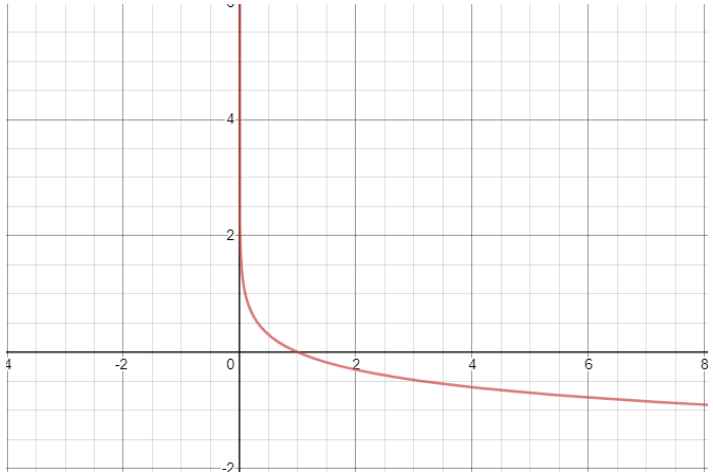
# 분류

- 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

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

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



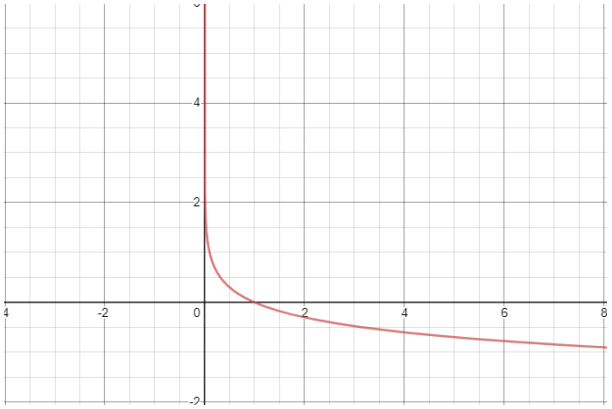



# 분류

- 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=∞

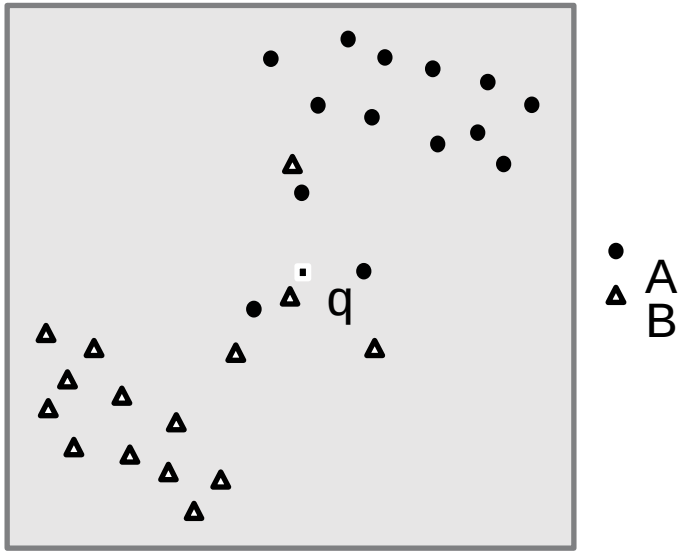


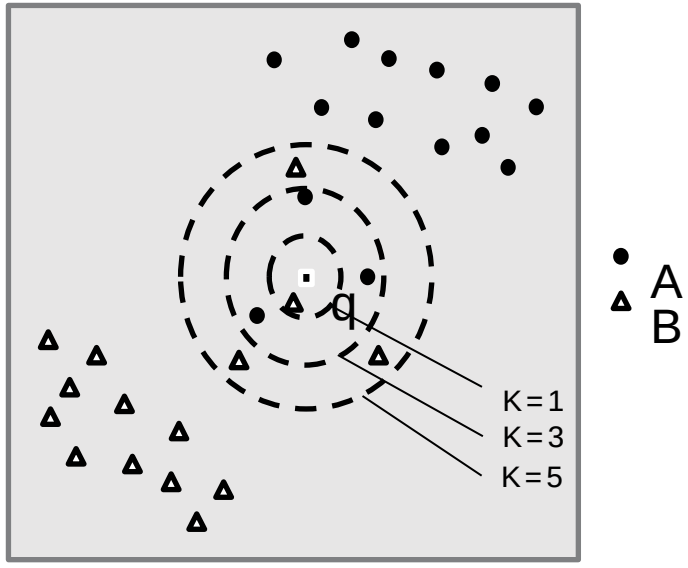

$$p_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} = \frac{e^{x_j}}{e^{x_1}+e^{x_2}+\cdots+e^{x_K}} \quad for\ j = 1,\ldots,K$$

...(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



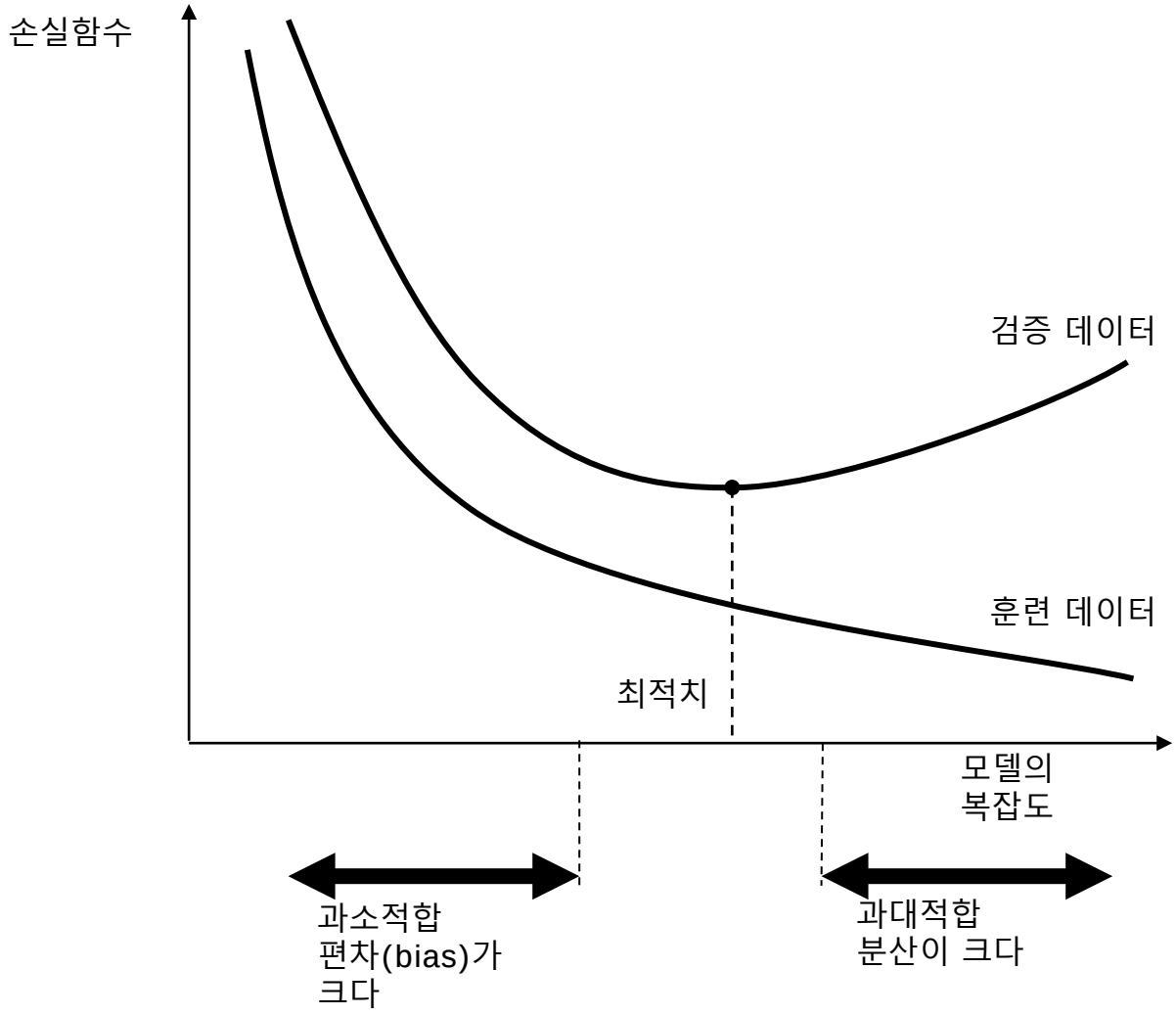



# 분류

- 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

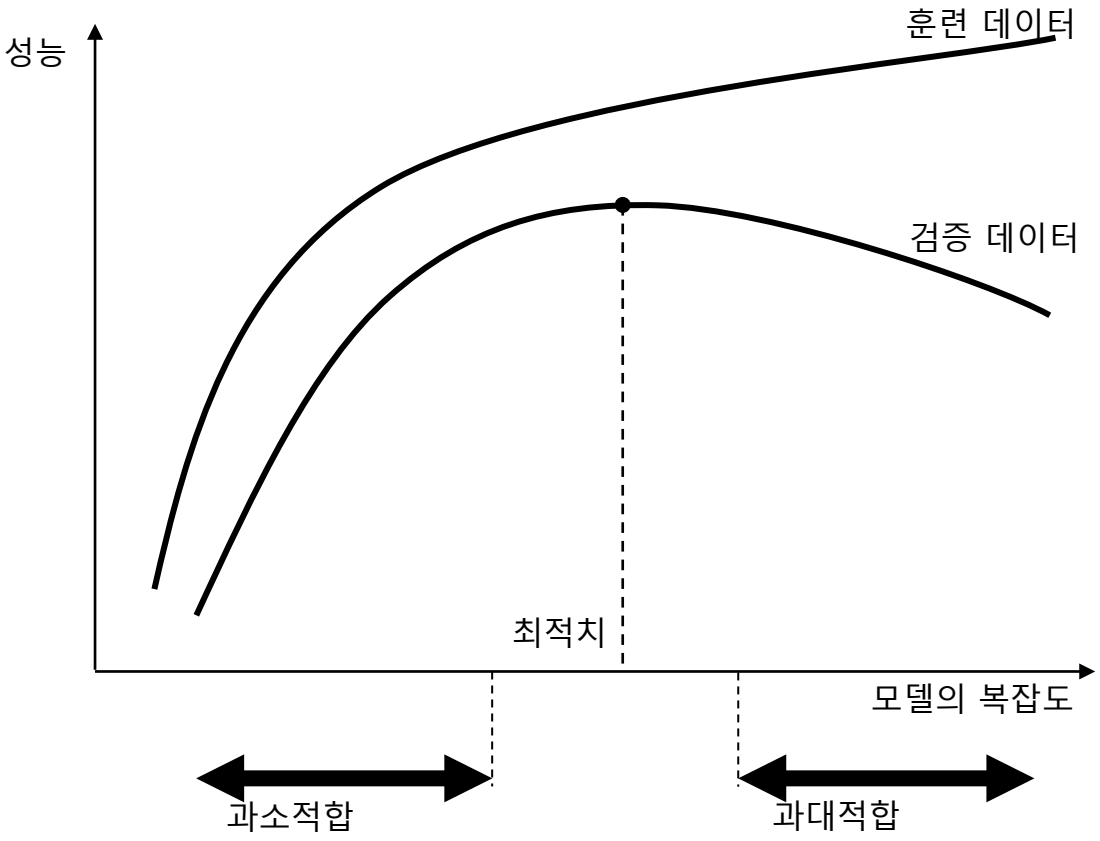
# 분류

- 손실함수 그래프로 과적합 판단



# 분류

- 성능 그래프로 과적합 판단



# 분류

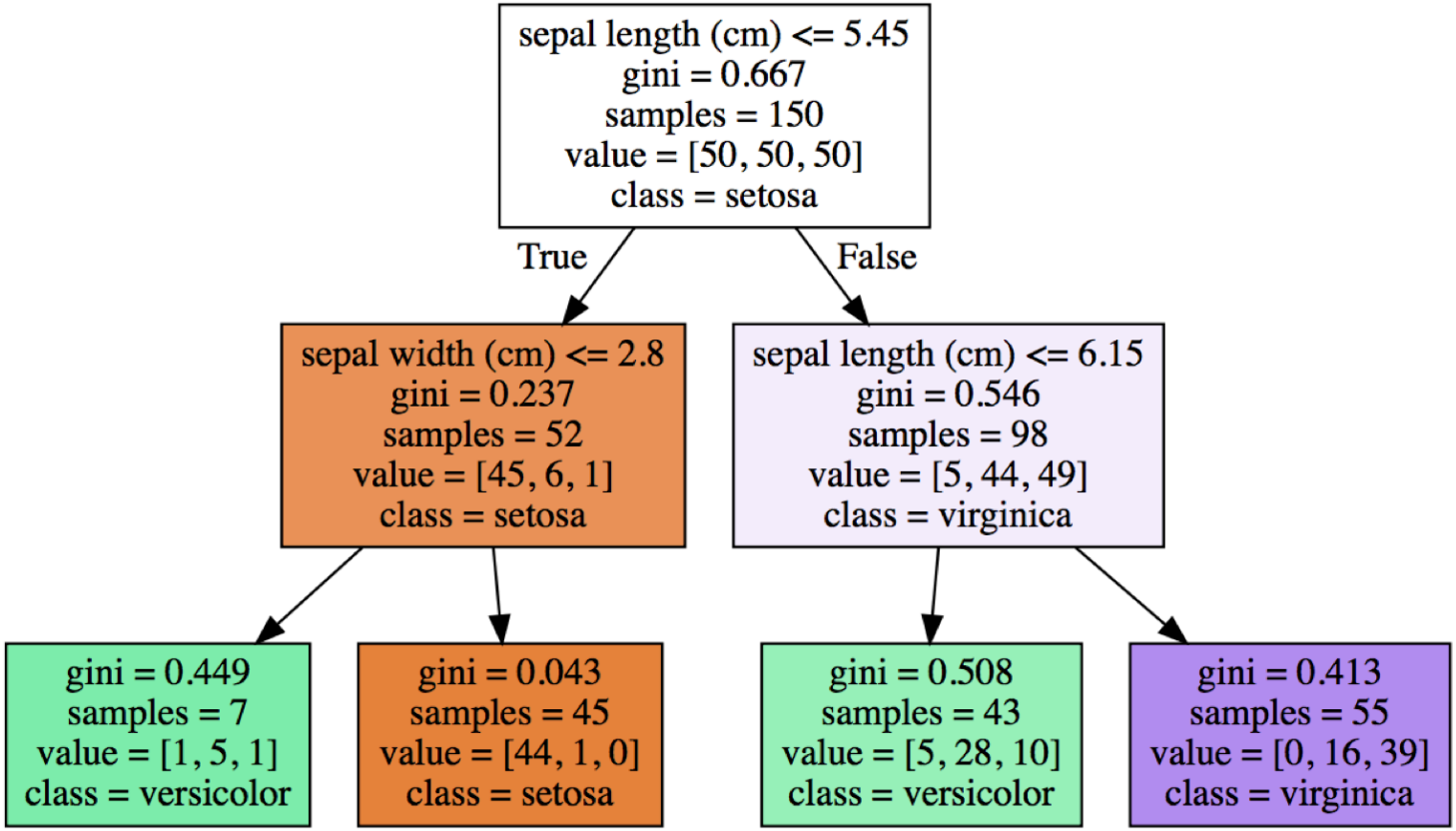
- 결정 트리
  - 분류 기법 적용에서 가장 많이 사용하는 방법
  - 분류 작업을 수행하기 위해 한번에 한 특성 변수를 해석
  - 결정 트리 모델을 사용하면 동작을 설명하기 수월함
    - 대출 거부 사유
    - 신용도가 낮은 이유
    - 불합격 사유 등
- 결정 트리 특징
  - 선형회귀 모델은 특성들을 대상으로 곱셈과 덧셈과 같은 연산을 하고 그 값을 기준으로 회귀나 분류를 예측했음
  - 결정 트리(decision tree)는 이와 달리 각 특성을 독립적으로 하나씩 검토하여 분류 작업을 수행함
    - 마치 스무고개 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

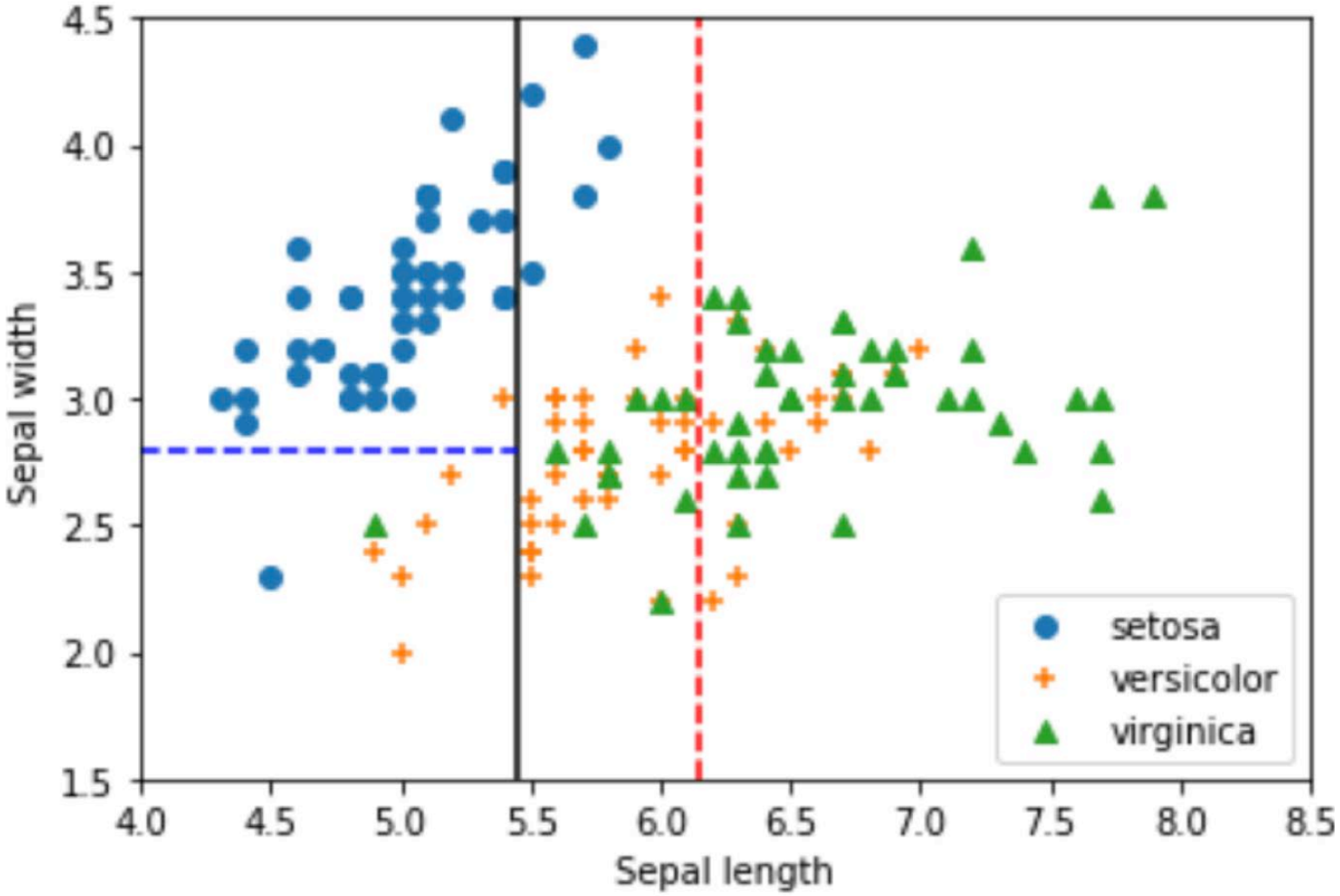
# 분류

- 결정트리 예


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

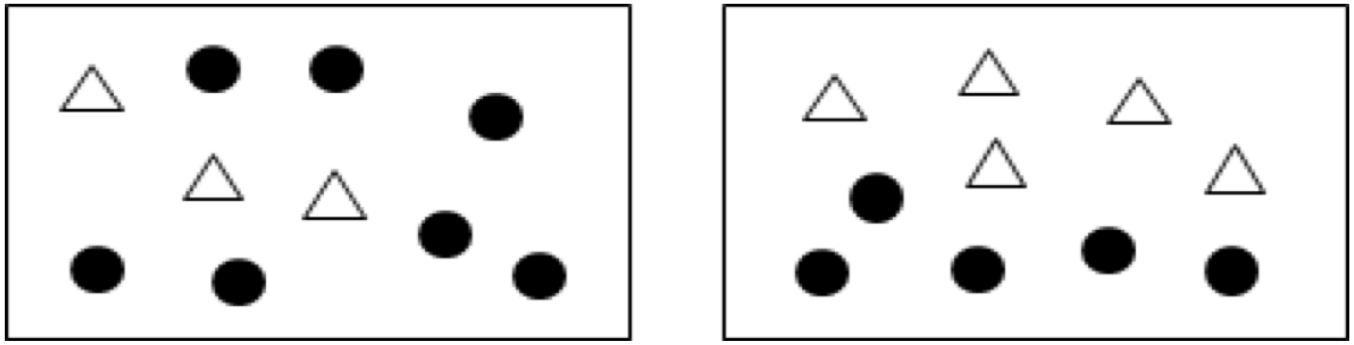
# 분류

- 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

- 종료 조건
  - 결정 트리를 계속 만들어 상세하게 분류를 하면 언젠가는 훈련 데이터에 대해서 100% 순도의 분류가 가능함
  - 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

- 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
  - 엔트로피의 정의는 아래와 같으며 위와 같은 분류시의 엔트로피를 구하면 아래와 같음

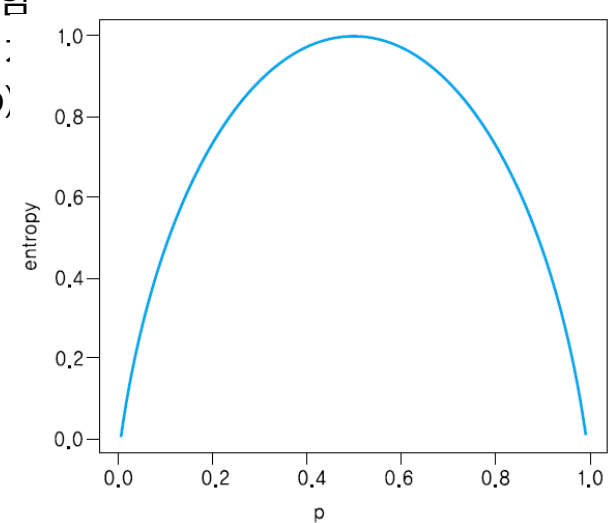
$$Entropy = -\sum_{k=1}^{m} p_k \log_2(p_k)$$

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

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

# 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 기대
    = p log(1/p) = -p log(p



# 분류

- 결정트리의 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
- 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

# 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
- 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: $\frac{1}{c}=\frac{(\frac{1}{a}+\frac{1}{b})}{2}$

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

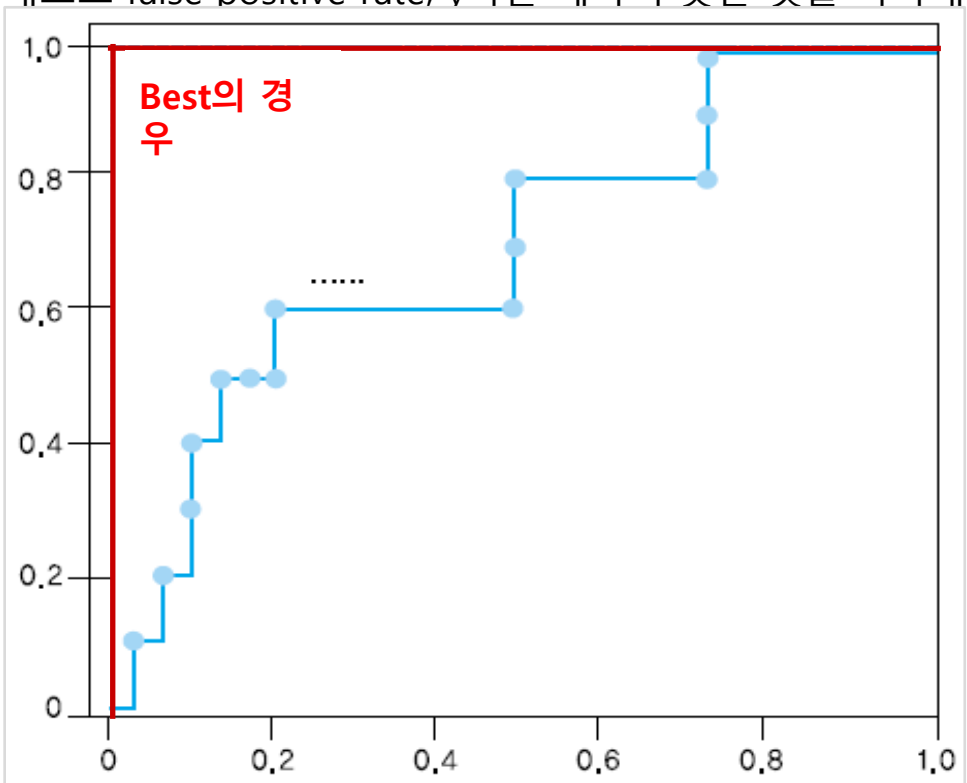
# 분류

- 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 |

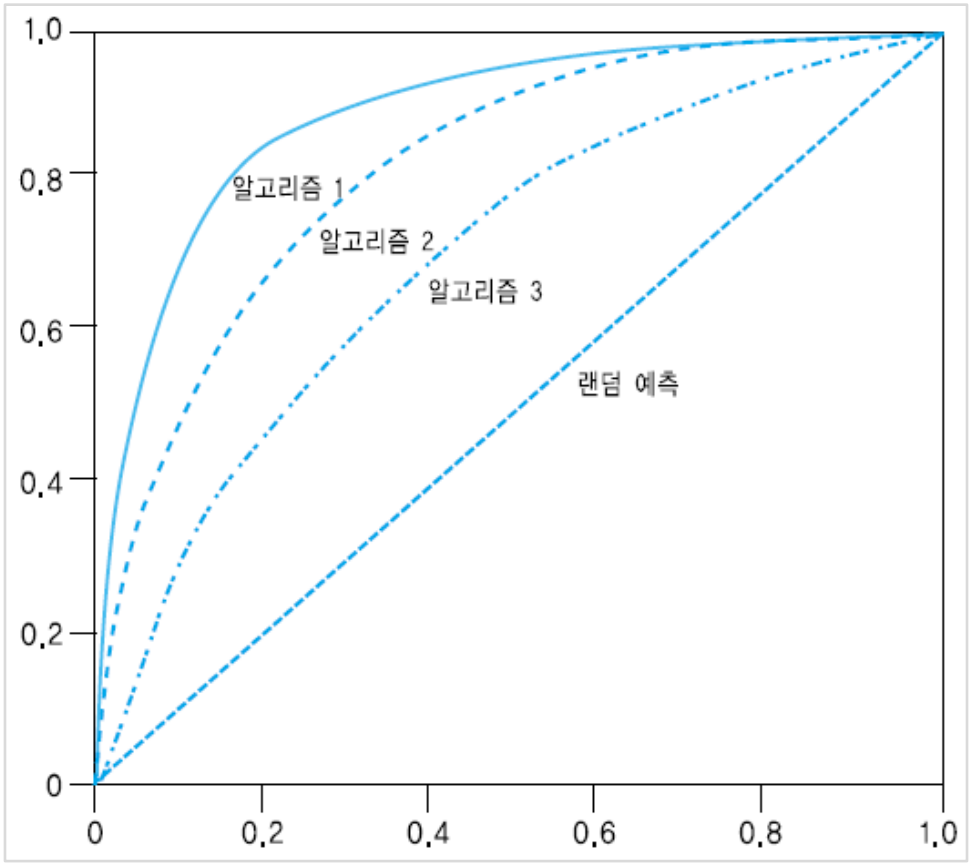
# 분류

- 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



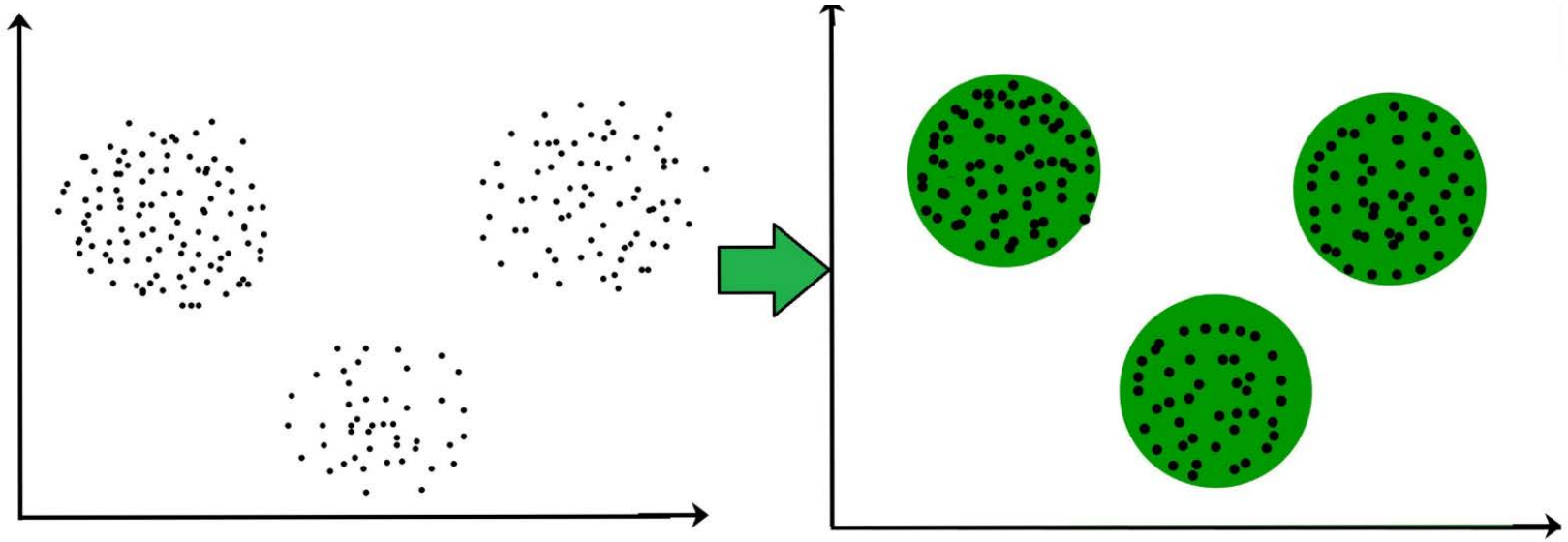
# 분류

- 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



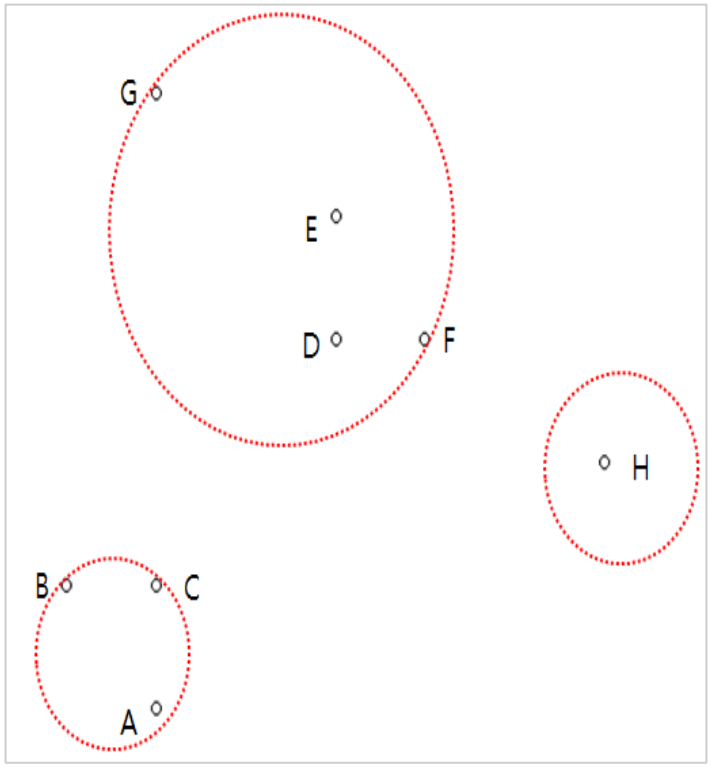
# 클러스터링

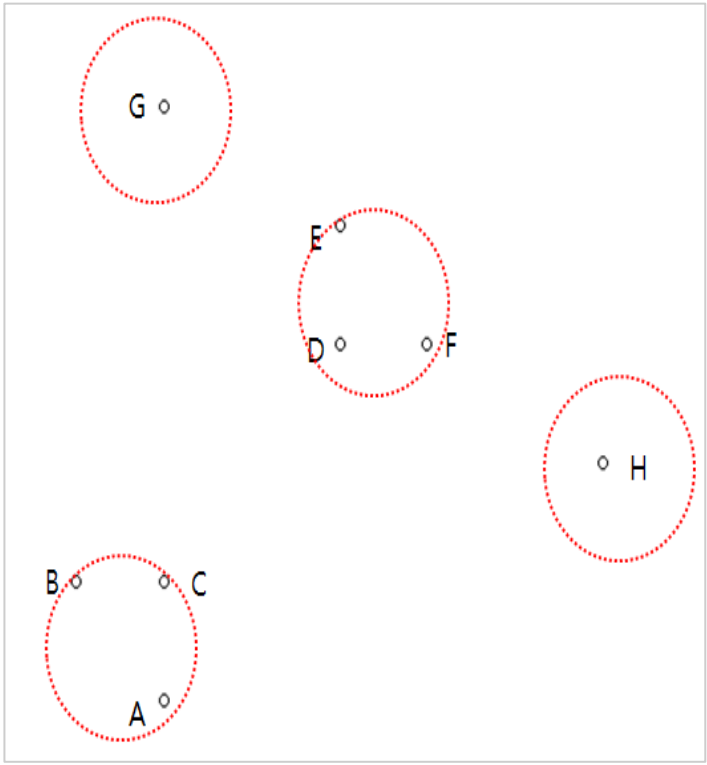
- 클러스터링
  - K개의 그룹에 속하는 유사한 샘플을 찾는 과정



# 클러스터링

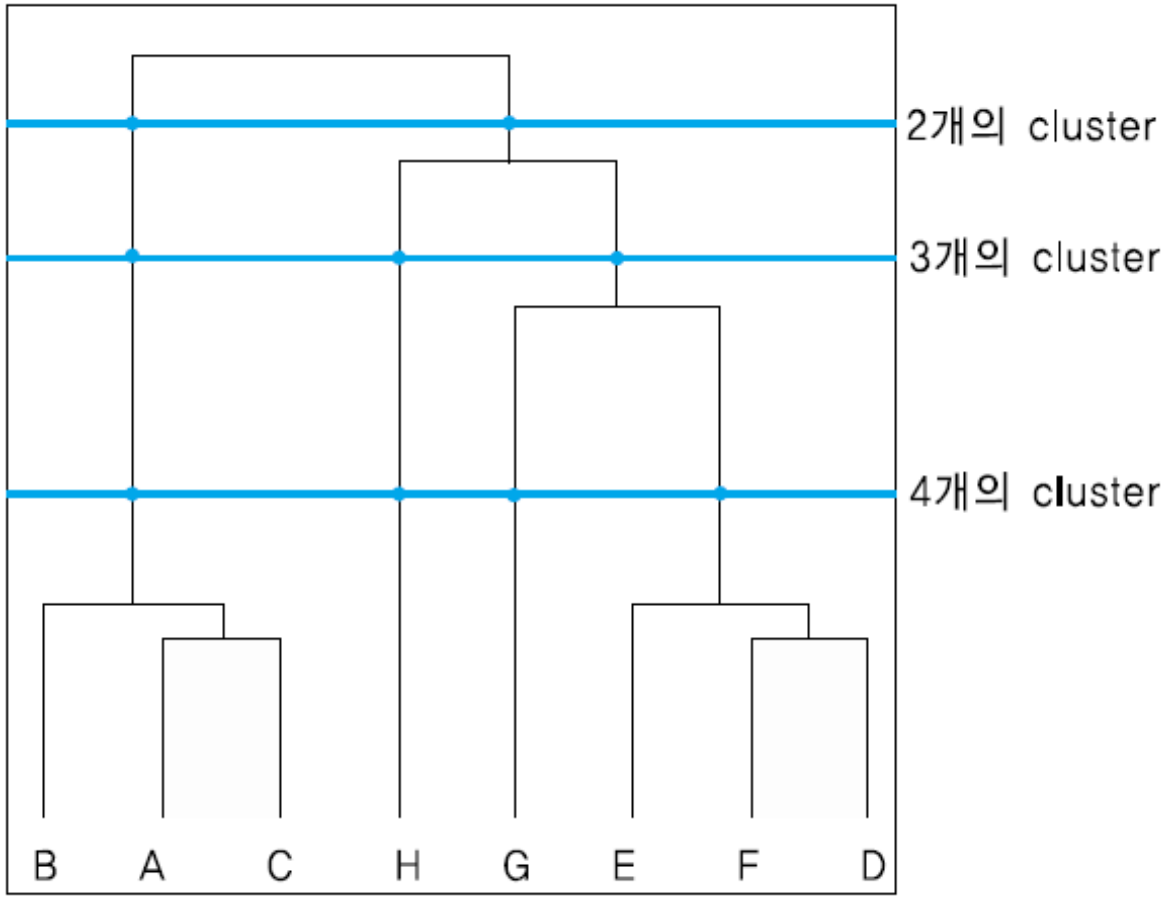
- 클러스터링
  - 적정한 군집의 수(k)를 먼저 찾아야 함



# 클러스터링

- 클러스터링
  - 적정한 군집의 수(k)를 먼저 찾아야 함



# 클러스터링

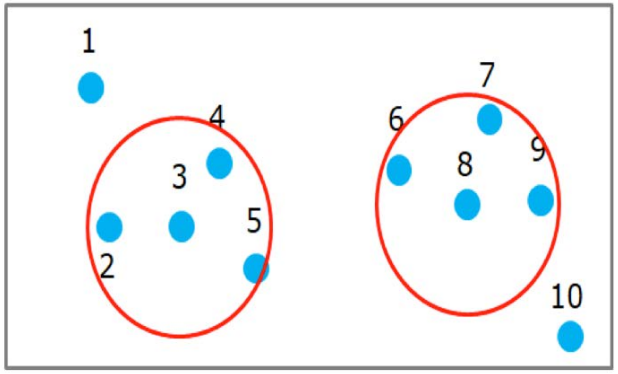
- 클러스터링 알고리즘
  - 조건
    - 같은 그룹 내의 항목들은 서로 속성이 비슷함 (유사도가 큼)
    - 다른 그룹에 속한 항목과는 속성이 서로 다름 (유사도가 작음)
  - 비정상 패턴 (이상치) 식별에도 사용된다
- K-means 알고리즘
  - 공간상에 임의의 k 개의 임의의 초기 지점을 클러스터 중점으로(cluster center) 정함
  - 클러스터 중점을 중심으로 거리가 가까운 항목을 선택하여 클러스터 공간을 나눔
  - 각 클러스터에 포함된 항목들의 평균 위치를 구해 이를 새로운 클러스터 중점(centroid)으로 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

- 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
  - 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

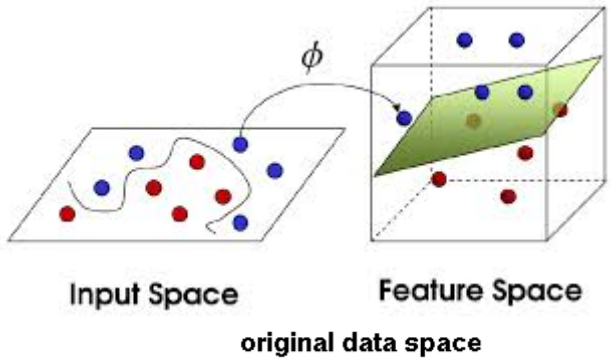
# 클러스터링

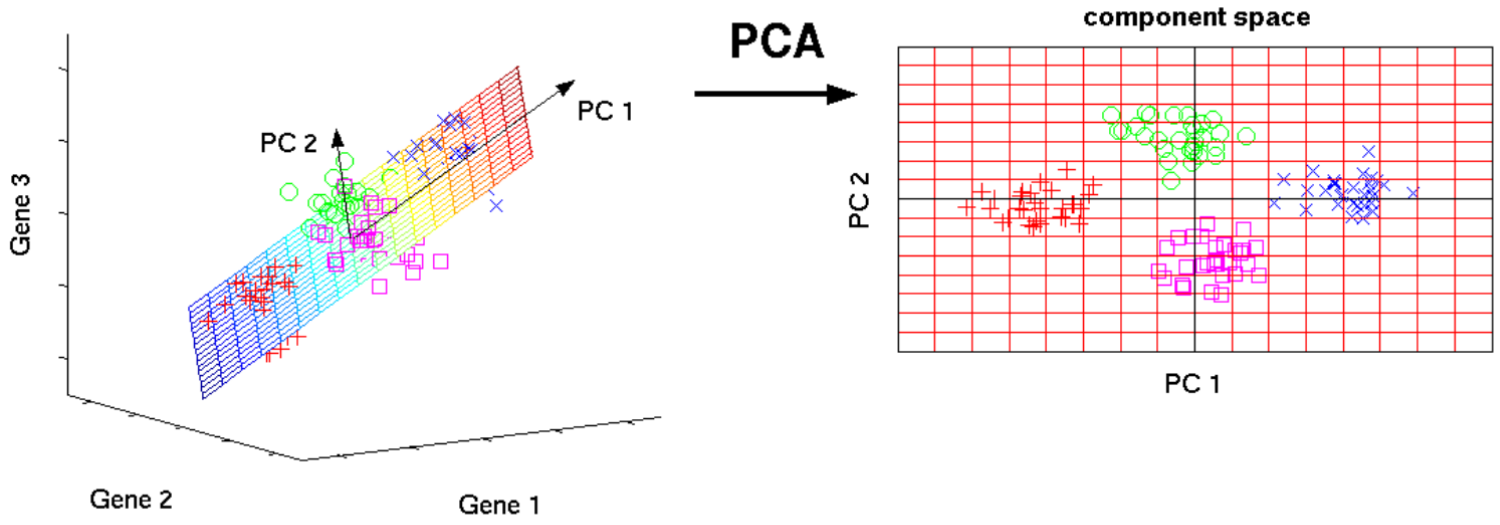
- DBSCAN
  - 밀도 기반 클러스터링 알고리즘
  - k-means처럼 단순히 거리만을 기준으로 군집화를 하는 것이 아니라 "가까이 있는 샘플들은 같은 군집에 속한다"는 원칙으로 군집을 차례로 넓혀가는 방식임
  - 샘플들의 몰려 있는 정도 즉, 밀도가 높은 부분을 중심으로 인접한 샘플들을 포함시켜 나감
  - 한 점을 기준점으로 반경 r내에 점이 n개 이상 있으면 하나의 군집으로 인식하는 방식임


# 차원 축소

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




Reduce unnecessary representation axis