2. 선형 회귀 모델

• INDEX

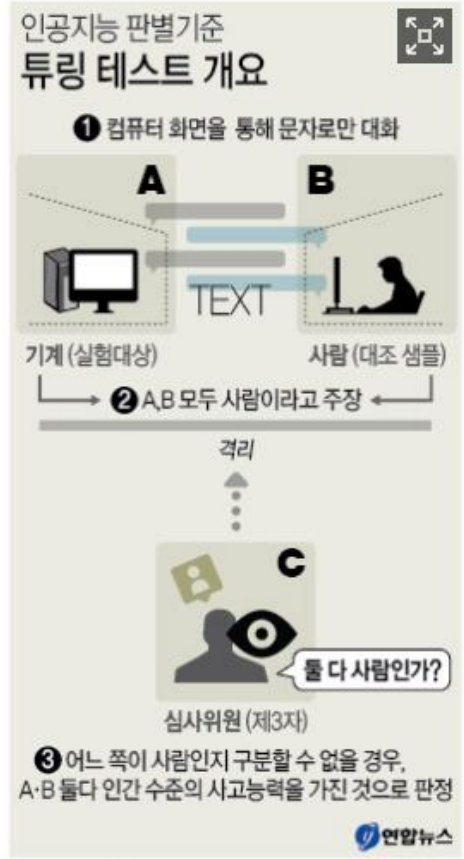
- 머신 러닝
- 선형 회귀
  - 정규 방정식
  - 경사 하강법
- 로지스틱 회귀
- 데이터 전처리

머신러닝

머신러닝

AI

- Artificial Intelligence
- 인간의 학습, 추론, 지각 능력 등을 컴퓨터를 통해 구현하는 가장 포괄적인 분야
- AI의 최종 목표는 기계가 사람처럼 보고, 듣고, 말하고, 생각하게 만드는 것
- AI를 구현하는 두 가지 접근법
  - 규칙 기반
    - 전문가의 지식과 규칙을 사람이 직접 컴퓨터에 코드로 입력하는 방식
    - 한계: 세상의 모든 규칙을 만들 수 없고, 예외 상황에 대처하기 어려움
  - 학습 기반(머신러닝)
    - 컴퓨터에게 데이터를 주고, 그 데이터 안에서 규칙(패턴)을 스스로 배우게 하는 방식
    - 현대 AI의 핵심 동력이며, 우리가 배울 분야



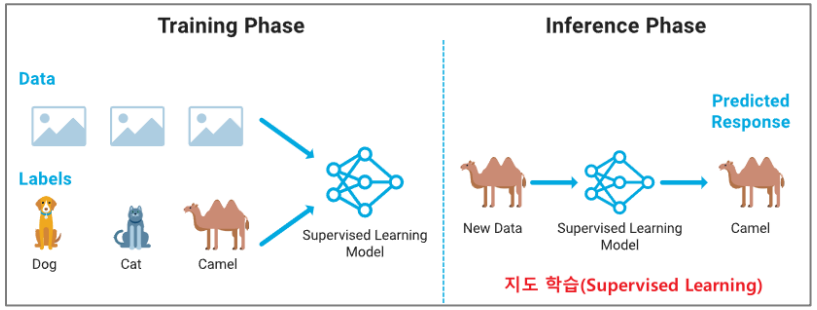
이재윤 기자 / 20140609
@yonhap_graphics(트위터)

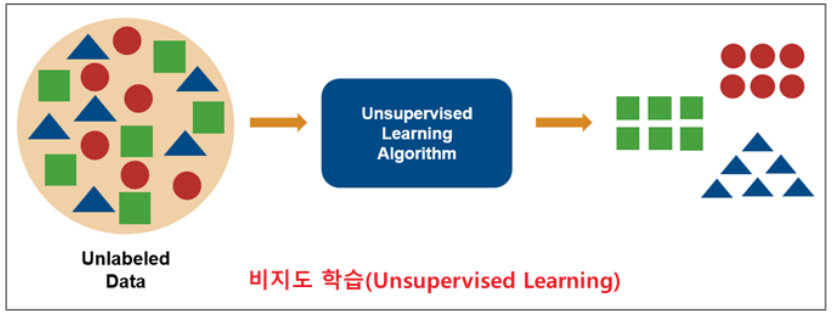
https://www.yna.co.kr/view/AKR20140609093300091

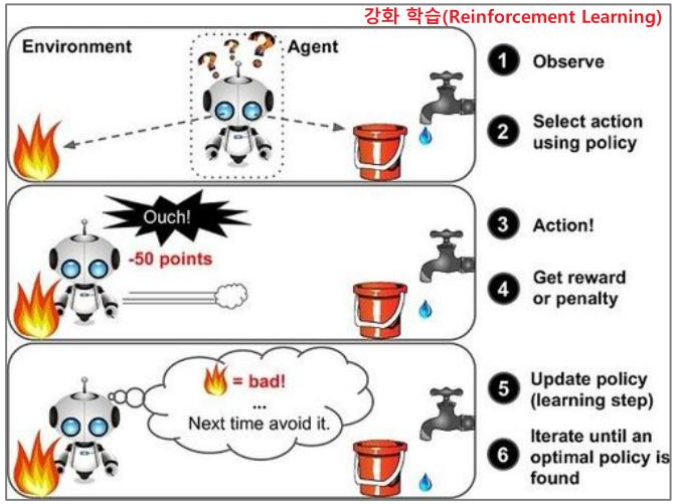
14기 서울18반 이재윤

머신러닝

- 컴퓨터에게 데이터를 주고, 그 데이터 안에서 규칙(패턴)을 스스로 학습하게 하는 방식
- 학습 종류
  - **지도 학습(Supervised Learning)**
    - 정답지가 있는 데이터를 학습시키는 방법
    - 의료 진단, 스팸 필터링, 주가/부동산 가격 예측 등에 활용
  - **비지도 학습(Unsupervised Learning)**
    - 정답이 없는 데이터의 숨겨진 구조나 규칙을 찾는 방법
    - 추천 시스템, 이상 거래 탐지, 고객 세분화 등에 활용
  - **강화 학습(reinforcement Learning)**
    - '상' 과 '벌' 을 통해서 최적의 행동 방식을 학습하는 방법 (알파고)
    - 로봇 제어, 자율 주행, 게임 AI 등에 활용
- 일반적으로 비지도 -> 지도 -> 강화 학습으로 복합적인 학습으로 결과물 생성





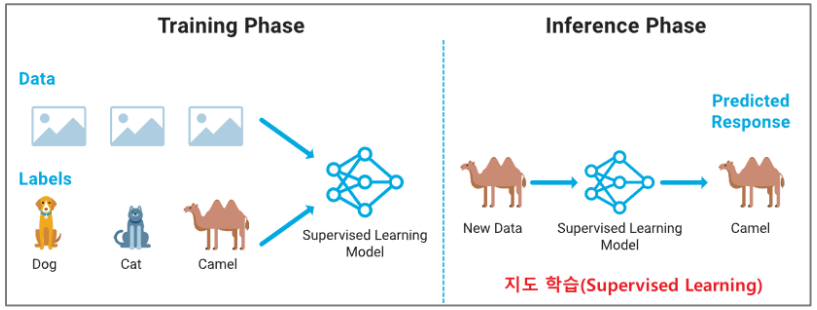


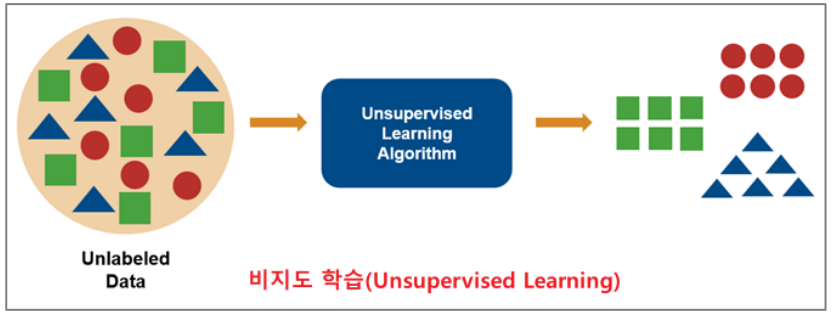
https://www.mathworks.com/discovery/supervised-learning.html?s_tid=srchtitle_site_search_1_supervised+learning
https://www.mathworks.com/discovery/unsupervised-learning.html https://marutitech.com/businesses-reinforcement-learning/

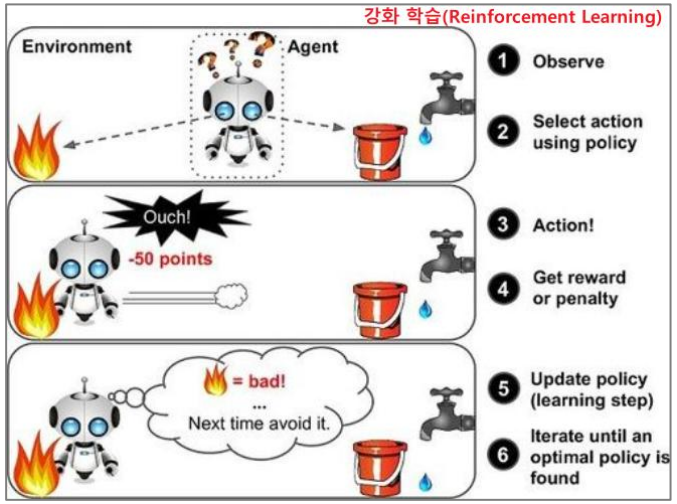
14기 서울18반 이재윤

머신러닝

- 컴퓨터에게 데이터를 주고, 그 데이터 안에서 규칙(패턴)을 스스로 학습하게 하는 방식
- 학습 종류
  - **지도 학습(Supervised Learning)**
    - 정답지가 있는 데이터를 학습시키는 방법
    - 의료 진단, 스팸 필터링, 주가/부동산 가격 예측 등에 활용
  - 비지도 학습(Unsupervised Learning)
    - 정답이 없는 데이터의 숨겨진 구조나 규칙을 찾는 방법
    - 추천 시스템, 이상 거래 탐지, 고객 세분화 등에 활용
  - 강화 학습(reinforcement Learning)
    - '상' 과 '벌' 을 통해서 최적의 행동 방식을 학습하는 방법 (알파고)
    - 로봇 제어, 자율 주행, 게임 AI 등에 활용
- 일반적으로 비지도 -> 지도 -> 강화 학습으로 복합적인 학습으로 결과물 생성







https://www.mathworks.com/discovery/supervised-learning.html?s_tid=srchtitle_site_search_1_supervised+learning
https://www.mathworks.com/discovery/unsupervised-learning.html https://marutitech.com/businesses-reinforcement-learning/

지도 학습(Supervised Learning)

- 정답(레이블, Label)이 있는 데이터(피쳐, Feature)를 학습시키는 방법
- 우리가 맞춰야 할 '정답'에 따라 두 가지의 큰 유형으로 나뉨
- 지도 학습 유형
  - 회귀(Regression)
    - 연속적인 숫자 값을 예측하는 문제
    - Feature 와 Label 을 기반으로 학습한 모델이 "새로운 Feature"의 Label을 예측하는 것
    - ex) 공부한 시간에 따른 시험 점수 예측하기
  - 분류(Classification)
    - 주어진 Feature가 어떤 그룹(카테고리)에 속하는 지 예측하는 문제
    - ex) 메일의 내용을 보고 '스팸' 인지, '정상'인지 판단하기

지도 학습(Supervised Learning)

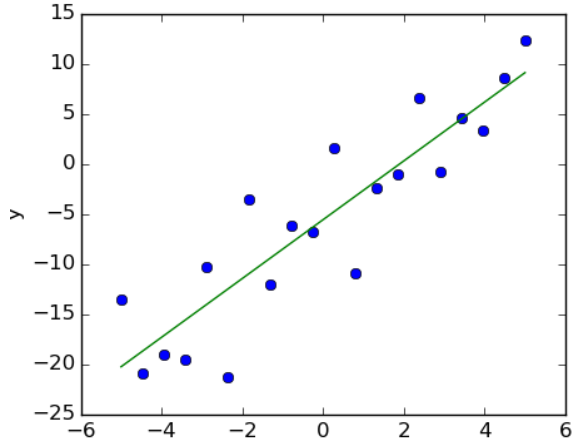
- 정답(레이블, Label)이 있는 데이터(피쳐, Feature)를 학습시키는 방법
- 우리가 맞춰야 할 '정답'에 따라 두 가지의 큰 유형으로 나뉨
- 지도 학습 유형
  - 회귀(Regression) => 회귀 문제를 푸는 가장 기본적이면서 강력한 도구가 "선형 회귀"
    - 연속적인 숫자 값을 예측하는 문제
    - Feature 와 Label 을 기반으로 학습한 모델이 "새로운 Feature"의 Label을 예측하는 것
    - ex) 공부한 시간에 따른 시험 점수 예측하기
  - 분류(Classification)
    - 주어진 Feature가 어떤 그룹(카테고리)에 속하는 지 예측하는 문제
    - ex) 메일의 내용을 보고 '스팸' 인지, '정상'인지 판단하기

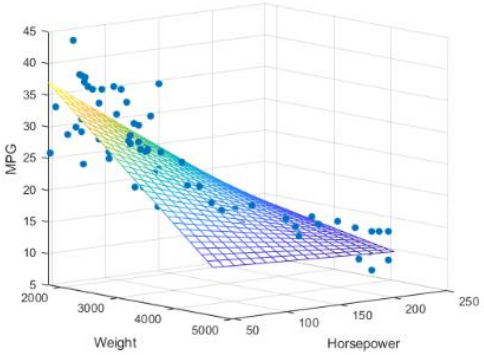
선형 회귀

14기 서울18반 이재윤

선형 회귀(linear regression)

- 예측을 "하나의 직선"으로 하는 가장 간단하고 직관적인 방법 (다차원일 경우에는 평면/초평면)
- (단순 선형 회귀) 하나의 특성(Feature)만 있을 때
  - 공부한 시간(x)에 따른 시험 점수(y) 예측
  - $y = wx + b$
  - w (가중치, Weight): 직선의 기울기. 공부 시간이 1시간 늘 때 점수가 몇 점 오르는지를 나타냄
  - b (편향, Bias): y절편. 공부를 하나도 안 했을 때의 기본 점수를 의미
- (다중 선형 회귀) 여러 특성이 있을 때
  - 캐럿(x_1), 투명도(x_2), 깊이(x_3) 를 모두 고려한 다이아몬드 가격(y) 예측
  - $y = w_1x_1 + w_2y_2 + w_3y_3 + \ \ldots + b$
  - 각각 의 특성에 대한 최적의 가중치와 편향을 찾는 것이 목표



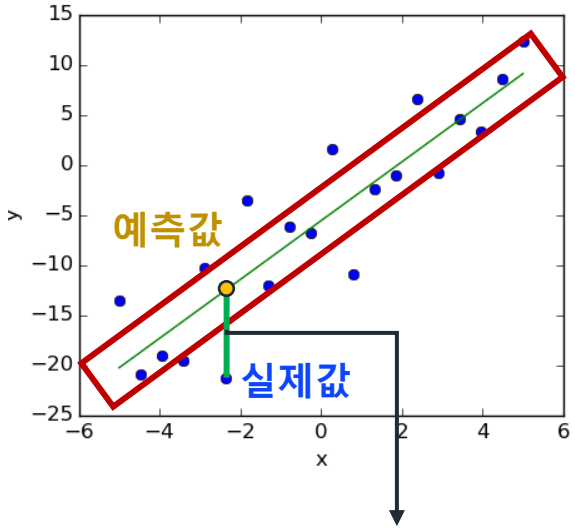


https://joshuajangblog.wordpress.com/2016/08/24/python-linear-regression/

선형 회귀

가설(Hypothesis)

- 우리가 예측에 사용할 "직선의 방정식(모델)"을 지칭하는 용어
- 그러면 어떤 가설(모델)이 좋은 가설일까?
  - 실제 데이터 들과의 거리(오차)가 가장 작은 직선
  - 이 값은 어떻게 구할까?



가설 = 모델

오차(예측값과 실제값의 차이)

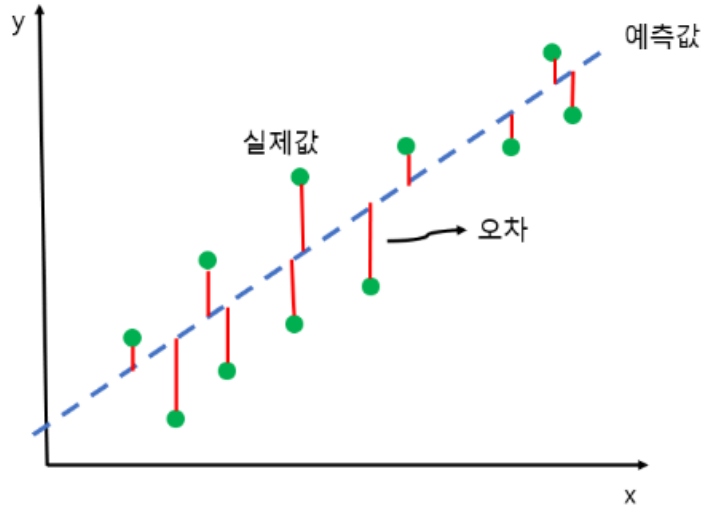
비용 함수(Cost Function)

- 모델(가설)이 얼마나 좋은 모델인 지 나타내는 점수 (like 시간복잡도)
- 엄연히 다르긴 하지만 손실 함수(loss function)라고도 함
- 비용 함수 종류
  - (회귀) 평균 제곱 오차(MSE, Mean Squared Error)
  - (회귀) 평균 절대 오차(MAE, Mean Absolute Error)
  - (분류) 교차 엔트로피
  - …

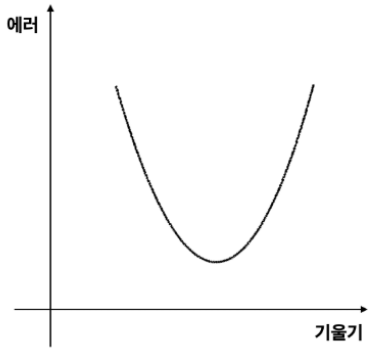
https://joshuajangblog.wordpress.com/2016/08/24/python-linear-regression/

평균 제곱 오차(MSE, Mean Squared Error) (1/2)

- 모델의 예측값이 실제 정답과 얼마나 차이 나는지 측정하는 방법
- 점수가 낮을수록 모델의 성능이 좋다는 의미
- 계산 방법
  1. 각 데이터들의 실제값과 예측값의 차이(오차)를 구한다.
  2. 각 데이터들의 오차를 모두 제곱하여 양수로 만들고 더하기
     (제곱하는 이유: 양수 오차와 음수 오차의 상쇄 막기 + 큰 오차에 페널티 부여)
  3. 제곱 총합을 데이터 개수로 나누어 평균을 구함



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



MSE 그래프

https://bigdaheta.tistory.com/85

https://curriculum.cosadama.com/machine-learning/3-2/

평균 제곱 오차(MSE, Mean Squared Error) (2/2)

- 예시) 시험 성적 예측 모델
  - **학생 A**: 1시간 공부해서 60점, **학생 B**: 2시간 공부해서 70점, **학생 C**: 3시간 공부해서 90점
  - **우리의 예측 모델 :** 예측 점수: 15 * (공부 시간) + 45 (15(가중치, w)와 45(편향, b)는 임의의 초기값)
  - 학생 A의 예측: 15 * 1 + 45 = 60 , 오차: 실제(60) - 예측(60) = 0
  - 학생 B의 예측: 15 * 2 + 45 = 75 , 오차: 실제(70) - 예측(75) = (-5)
  - 학생 C의 예측: 15 * 3 + 45 = 90 , 오차: 실제(90) - 예측(90) = 0
  - 각 오차를 제곱하여 더한 뒤, 학생 수로 나누기: ($0^2 + (-5)^2 + 0^2$) / 학생 수(3) = 25 / 3 = 8.33…

❖ 결국 우리의 목표는 **MSE 점수(비용)를 0에 가장 가깝게 만드는 최적의 모델(w와 b)을 찾는 것!!!!!!**

=> 이 원리가 **최소제곱법**(MLS, Method of Least Squares)

평균 제곱 오차(MSE, Mean Squared Error) (2/2)

- 예시) 시험 성적 예측 모델
  - **학생 A**: 1시간 공부해서 60점, **학생 B**: 2시간 공부해서 70점, **학생 C**: 3시간 공부해서 90점
  - **우리의 예측 모델 :** 예측 점수: 15 * (공부 시간) + 45 (15(가중치, w)와 45(편향, b)는 임의의 초기값)
  - 학생 A의 예측: 15 * 1 + 45 = 60 , 오차: 실제(60) - 예측(60) = 0
  - 학생 B의 예측: 15 * 2 + 45 = 75 , 오차: 실제(70) - 예측(75) = (-5)
  - 학생 C의 예측: 15 * 3 + 45 = 90 , 오차: 실제(90) - 예측(90) = 0
  - 각 오차를 제곱하여 더한 뒤, 학생 수로 나누기: ($0^2 + (-5)^2 + 0^2$) / 학생 수(3) = 25 / 3 = 8.33…

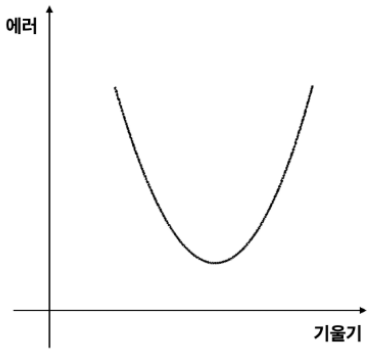
❖ 결국 우리의 목표는 **MSE 점수(비용)를 0에 가장 가깝게 만드는 최적의 모델(w와 b)을 찾는 것!!!!!!**

최소 제곱법(MLS, Method of Least Squares)

- 예측 모델($y = wx + b$)에서 비용*을 최소화하는 **w(가중치)** 와 **b(편향)**을 찾는 원리
- 그러면 비용을 어떻게 최소화해서 최적의 모델을 찾을까?
- 최적의 w 와 b 를 찾는 방법
  1. **정규 방정식(Normal Equation)**
     - 복잡한 미분 방정식을 풀어서 한 번에 정답을 계산해내는 방법
     - 만들어진 수학공식을 이용해, 최적의 파라미터를 한 번에 계산
  2. **경사 하강법(Gradient Descent)**
     - 최적의 파라미터를 점진적으로 찾아 나가는 방법
     - 대부분의 모델이 이 방식을 채택

* 비용은 최소 제곱 오차(MSE)를 채택

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



MSE 그래프

정규 방정식

정규 방정식 (1/2)

- 복잡한 미분 방정식을 풀어서 한 번에 정답을 계산해내는 방법
- 만들어진 수학공식을 이용해, 최적의 파라미터를 한 번에 계산

$$\theta = (X^T X)^{-1} X^T y$$

- θ (세타): 찾으려는 최종 목표, 모든 가중치(W)와 편향(b)을 담고 있는 결과값
- X: 입력 데이터(feature) 행렬, y: 맞춰야 할 정답(레이블) 벡터
- 예시) 시험 성적 예측 모델
  - **학생A**: 1시간 공부해서 60점, **학생B**: 2시간 공부해서 70점, **학생C**: 3시간 공부해서 90점

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \quad y = \begin{bmatrix} 60 \\ 70 \\ 90 \end{bmatrix} \quad \theta = \begin{bmatrix} b \\ w \end{bmatrix} = \begin{bmatrix} 43.33 \\ 15 \end{bmatrix}$$

  - 예측 점수 = 15 * (공부 시간) + 43.33

14기 서울18반 이재윤

정규 방정식 (2/2)

전치 역행렬

$$\theta = (X^T X)^{-1} X^T y$$

- 장점
  - 데이터의 특성(feature)가 적을 때, 빠르고 정확하게 해를 찾음
  - 추가 설정이 필요없음
- 단점
  - 데이터의 특성(feature)가 많으면 속도가 매우 느려짐
  - 역행렬이 존재하지 않으면 계산할 수 없음
  - 역행렬 계산 시 미세한 오차로 인해 계산 결과가 완전히 틀어질 수 있음

➢ 이런 역행렬이 존재하지 않으면 계산할 수 없는 문제는 "특이값 분해(SVD)" 로 해결

❖ 특이값 분해(SVD)

- 역행렬이 없는 행렬을 역행렬이 있는 가장 유사한 행렬을 만들어내는 것
- 깊은 내용이니 진행 X

경사 하강법

경사 하강법 (Gradient Descent) (1/2)

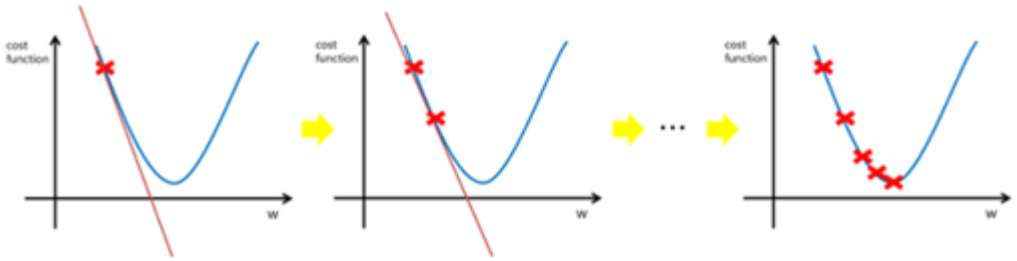
- 최적의 파라미터를 점진적으로 찾아 나가는 방법 * 비용은 최소 제곱 오차(MSE)를 채택
- 비용 함수의 기울기가 0 이 되는 지점이 가장 비용이 낮은 지점
- 동작 과정
  1. 랜덤한 지점에서 시작해서 현재 지점의 "기울기"를 구한다.
  2. 해당 기울기만큼 움직인다.
     (움직이는 보폭을 "학습률(Learning Rate)" 라고 함)
  3. 이동한 위치에서 다시 경사를 확인하고, 학습률만큼 이동한다.
  4. 위 과정을 계속 반복하면서, 가장 낮은 지점(기울기가 0)에 도달

[참고]기울기 구하는 방법

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

미분

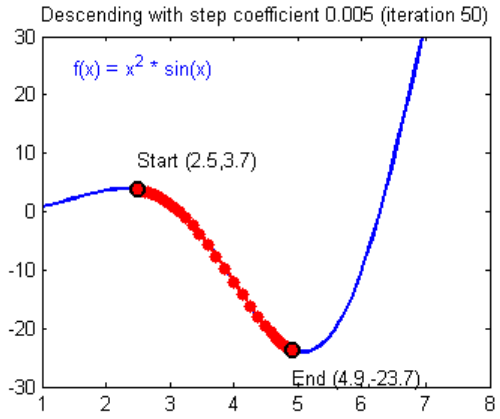
$$\nabla_\theta = \frac{2}{m}X_b^T(X_b\theta - y)$$

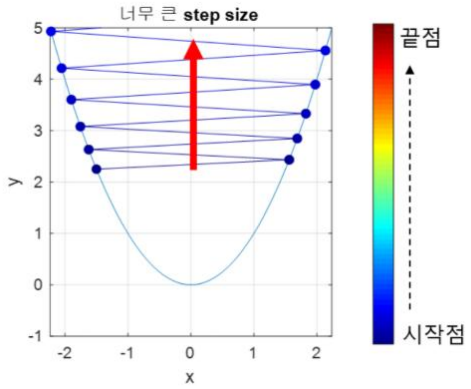


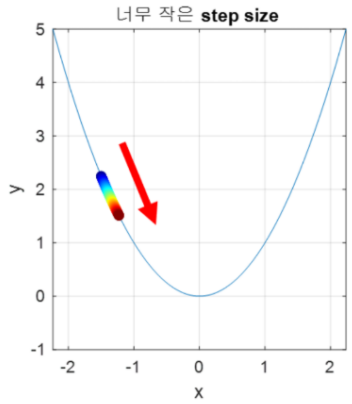
https://m.blog.naver.com/sqlmvp/221855713144

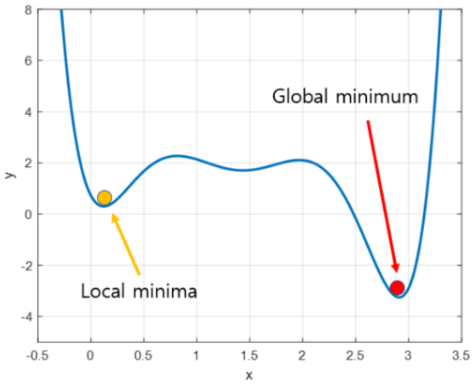
경사 하강법 (Gradient Descent) (2/2)

- 기울기만큼 이동(학습률)할 때, 반대 방향으로 이동함 (음수 기울기면, 양수 방향으로 이동해야 함)
- 학습률 설정
  - 학습률이 너무 크면, 보폭이 너무 커서 반대편으로 올라갈 수 있음 (발산, overshooting)
  - 학습률이 너무 작으면, 너무 오래 걸림
- 지역 극솟값(local minima) 문제: 시작 위치가 랜덤이기 때문에 Global minimu이 아닌 local minimum에 빠질 수 있음









https://twojun-space.tistory.com/124

경사 하강법 (Gradient Descent)에서의 용어/파라미터 (1/2)

- **기울기(gradient)**
  - **전체 데이터**가 만들어내는 "평균적인 오차"의 기울기
- **이터레이션(iterations)**
  - 데이터의 **일부** 혹은 **전체**를 보고, 현재 위치의 경사를 계산하고, **한 걸음** 이동하는 과정
- **에포크(Epoch)**
  - 전체 데이터를 모두 한 번 훑어보면, 1 에포크
- **학습률(Learning Rate, alpha)**
  - 한 번에 얼마나 크게 움직일지를 결정하는 보폭

경사 하강법 (Gradient Descent)에서의 용어/파라미터 (2/2)

- **허용 오차(Tolerance, tol)**
  - 학습 조기 종료 조건
  - 모델의 오차 감소량이 매우 작아지면, 학습해도 성능향상이 없겠다고 판단하여 학습을 멈춤
- **배치 크기(batch)**
  - 기울기를 구할 때, 전체 데이터 오차의 기울기를 한 번에 구하지 않음 => 너무 느림
  - 작은 묶음(배치)로 나눠서 처리
  - 전체 데이터 오차의 기울기를 한 번에 구하는 경우에는 "1 이터레이션" = "1 에포크"
- **기울기 누적 스텝(Gradient Accumulation Steps)**
  - 실질적인 배치 크기를 늘리는 고급 기법
  - 매 배치마다 업데이트하는 것이 아닌, n 번의 배치 간격으로 누적한 기울기를 기반으로 이동

경사 하강법

경사 하강법 (Gradient Descent)의 종류

1. **배치 경사 하강법(Batch Gradient Descent)**
   - 전체 데이터를 훑어보고, 기울기를 계산
   - 1 epoch = 1 iteration
   - 정확하지만, 데이터가 크면 매우 느림
2. **확률적 경사 하강법(SGD , Stochastic Gradient Descent)**
   - 하나의 데이터만 보고 바로 기울기를 계산
   - 매우 빠르지만, 불안
3. **미니배치 경사 하강법(Mini-batch Gradient Descent)**
   - 적당한 수의 배치(batch)만 보고, 기울기를 계산
   - 속도와 안정성을 모두 잡은 방법
   - 배치 수는 정해진 답이 없으며, 실험적으로 구해야 함

로지스틱 회귀

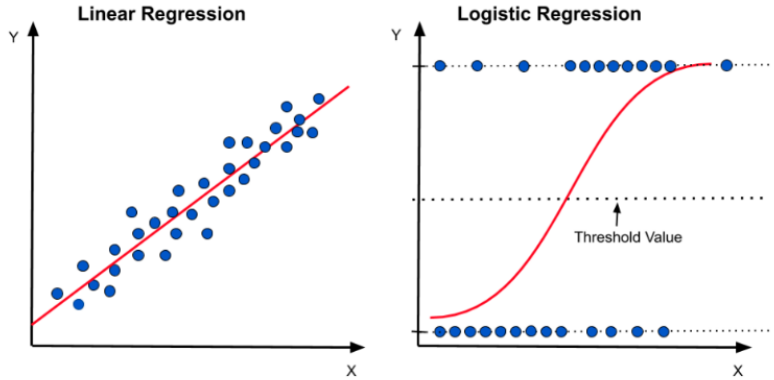
로지스틱 회귀

지도 학습(Supervised Learning)

- 정답(레이블, Label)이 있는 데이터(피쳐, Feature)를 학습시키는 방법
- 우리가 맞춰야 할 '정답'에 따라 두 가지의 큰 유형으로 나뉨
- 지도 학습 유형
  - 회귀(Regression)
    - 연속적인 숫자 값을 예측하는 문제
    - Feature 와 Label 을 기반으로 학습한 모델이 "새로운 Feature"의 Label을 예측하는 것
    - ex) 공부한 시간에 따른 시험 점수 예측하기
  - 분류(Classification) => 분류 문제를 푸는 가장 기본적이면서 강력한 도구가 "로지스틱 회귀"
    - 주어진 Feature가 어떤 그룹(카테고리)에 속하는 지 예측하는 문제
    - ex) 메일의 내용을 보고 '스팸' 인지, '정상'인지 판단하기

근데 그냥 선형 회귀로 분류 문제를 풀면 안되나요? 네, 안됩니다. (단호) (1/3)

1. 예측값이 범위를 벗어남
   - 선형 회귀는 끊임없이 이어지는 직선으로 되어 있고, 어떠한 입력값이든 비례하는 값을 예측함
   - 이는 분류 범위을 훨씬 벗어나는 예측값을 계산하게 됨
   - 예를 들면, 환자가 암에 걸릴 확률이 150% or -230% 는 의미가 없음

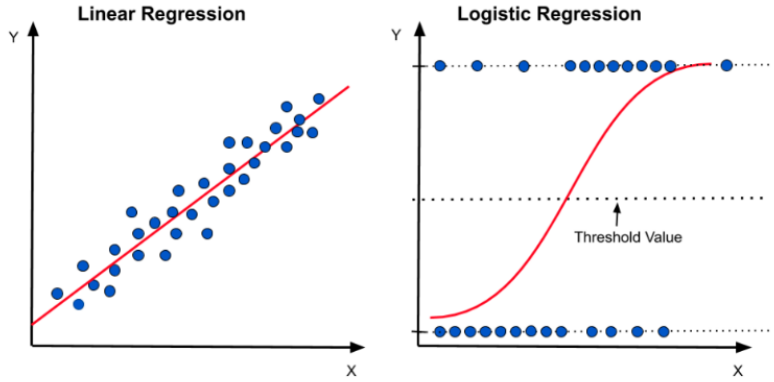


✓ 100% 이상은 100%, 0% 이하는 0% 식으로 결과만 자르면 안되나요 ?
   - 임시 방편에 불과함
   - 마치 0점, 1점 과녁에서 1점 방향으로 쐈다고 1점을 주는 것과 같음

근데 그냥 선형 회귀로 분류 문제를 풀면 안되나요? 네, 안됩니다. (단호) (2/3)

2. 이상치(Outlier)에 매우 민감

- 선형 회귀는 모든 데이터와의 평균적인 거리를 최소화하는 것이 목적
- 이상치가 추가된다면, 이상치에 따라 직선의 기울기가 바뀌게 됨
- 이로 인해서 "분류 기준선(Decision Boundary)"가 완전히 망가지는 현상이 발생



✓ 회귀 문제에서도 이상치가 나오면 동일한 문제가 발생하는 거 아닌가요 ?

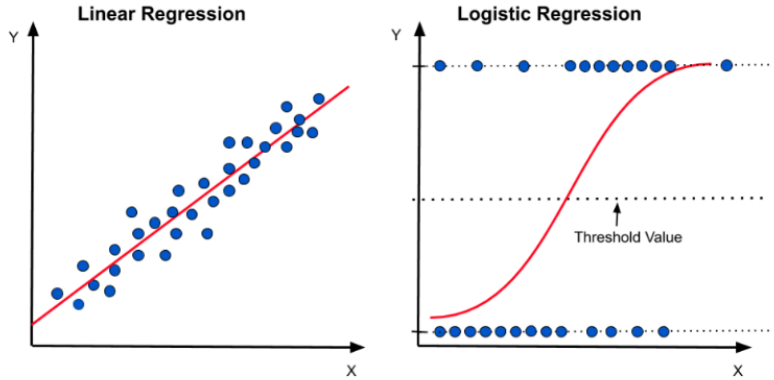
- 회귀에서는 단지 "하나의 예측을 크게 틀리게 만드는 것" 정도이지만,
- 분류에서는 "전체 규칙 자체를 바꿔버림"

14기 서울18반 이재윤

근데 그냥 선형 회귀로 분류 문제를 풀면 안되나요? 네, 안됩니다. (단호) (3/3)

3. 다중 분류 문제에서 잘못된 관계를 학습

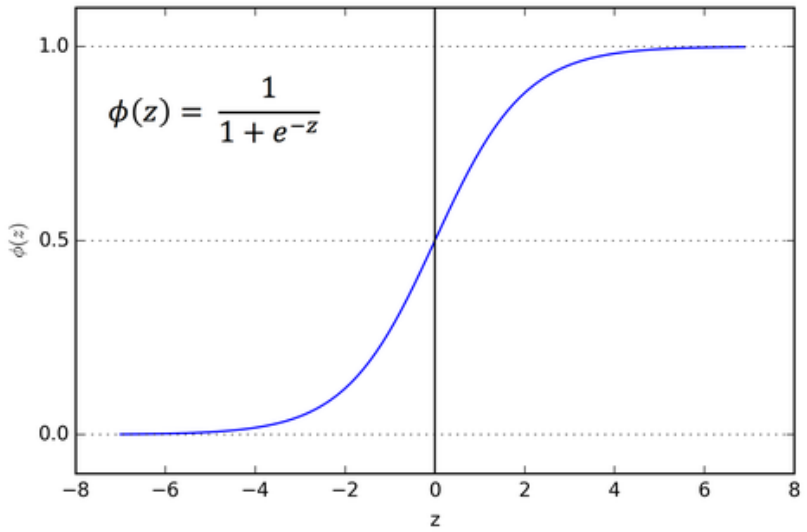
- 분류할 그룹이 3개 이상일 경우에 그룹들간에 수학적인 관계가 있다고 착각함
- 예) 3개의 그룹을 분류("개", "고양이", "새")
  - 분류를 위해 개=1, 고양이=2, 새=3 으로 설정
  - "개(1)" 와 "새(3)" 의 중간은 "고양이(2)" 와 같이
    실제로는 존재하지 않는 거짓된 순서나 크기 관계를 학습하게 됨



- 그러면 위의 문제(1.예측값 오류, 2.이상치, 3.다중 분류 문제)를 어떻게 해결할까?
  - 선형 회귀의 예측값을 "0과 1사이의 값으로 바꿔서" 생각하자 . (1번, 2번 문제 해결 가능)
  - "시그모이드(Sigmoid)" 함수의 등장!!

시그모이드(Sigmoid) 함수

- 어떤 숫자든 입력받아 그 값을 0과 1 사이의 값으로 압축해주는 S자 모양의 함수
- 음의 무한대부터 양의 무한대까지 어떤 숫자든 넣을 수 있고, 결과는 항상 0과 1사이의 값으로 나옴
- 공식의 특징
  - z가 아주 큰 양수이면, 분모가 0에 가까워져서 결과는 1에 가까워 짐
  - z가 아주 큰 음수이면, 분모가 무한대에 가까워져서 결과는 0에 가까워 짐



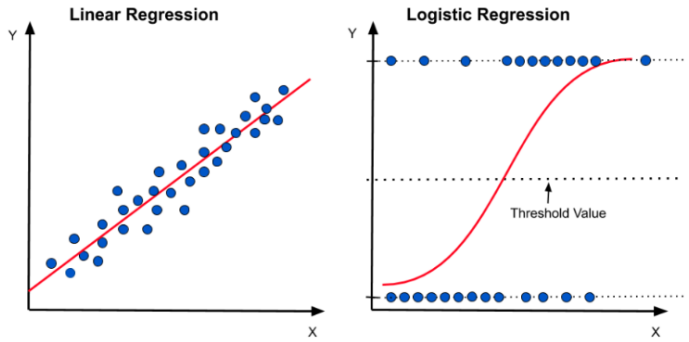
- 한계:
  - 이후에 배울 역전파에서 기울기를 전달하는 과정에서 기울기가 없어지는, 기울기 소실(Vanishing Gradient) 문제가 발생할 수 있음
  - ➢ ReLU 함수와 같은 다른 활성화 함수들이 더 널리 사용됨 (이후 학습 예정)

https://datasciencebeehive.tistory.com/80

14기 서울18반 이재윤

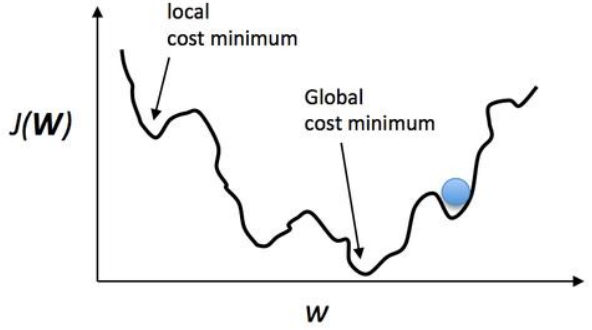
로지스틱 회귀(Logistic Regression)

- 선형 회귀 + 시그모이드 함수
- 데이터를 특정 그룹으로 나누는 분류 방법(이진 분류)
- 기존 채점 기준인 MSE(평균 제곱 오차) 대신 "이진 교차 엔트로피(BCE)"를 사용



❖ 평균 제곱 오차를 사용하지 못하는 이유는?

1. 경사 하강법이 잘 작동하려면 비용 함수 모양이 매끄러워야 하는데,
   시그모이드는 울퉁불퉁한 모양을 만들어 냄.
   이는 수 많은 **지역 최저점(Local minima)**으로 학습에 어려움을 줌



2. 모델이 학습에 게을러지는데, 정답이 "1" 이고, 예측값이 "0.01" 이라고 하면 아주 큰 실수이지만,
   정작 모델은 작은 예측값에 따라, 기울기도 0에 가까워지고, 그로 인해 움직이지 않는 "**기울기 소실 문제**"가 발생

로지스틱 회귀

이진 교차 엔트로피(BCE, Binary Cross-Entropy)

- 두 가지 중 하나의 답을 맞혀야 하는 이진 분류 문제를 위한 전용 손실 함수

$$L = -[y \log(\hat{p}) + (1 - y) \log(1 - \hat{p})]$$

- 정답이 1인 경우
  - 모델이 1에 가까운 확률을 예측할수록 0에 가까운 손실을 줌 (잘함)
  - 모델이 0에 가까운 확률을 예측할수록 무한대에 가까운 손실을 줌(못함)
- 정답이 0인 경우
  - 모델이 0에 가까운 확률을 예측할수록 0에 가까운 손실을 줌(잘함)
  - 모델이 1에 가까운 확률을 예측할수록 무한대에 가까운 손실을 줌(못함)

- 결론: **틀린 답에 대해, 모델이 얼마나 강하게 확신했느냐에 따라 벌점의 크기를 기하급수적으로 늘리는 것**

로지스틱 회귀

로지스틱 회귀(Logistic Regression) 완성

- 데이터를 특정 그룹으로 나누는 분류 방법(이진 분류)
- 선형 회귀 + 시그모이드 함수 + 이진 교차 엔트로피(BCE)
- 이후 학습은 선형 회귀와 마찬가지로 "**경사 하강법**"을 사용하고, 이진 교차 엔트로피(BCE) 비용을 최소화하는 방향으로 w와 b를 계속 업데이트 함

- 그러면 아까 "시그모이드 함수"로도 해결 못 한 "**다중 분류 문제**"는 어떻게 해결할까?

➢ 시그모이드 함수 대신 "소프트맥스(Softmax)" 를 활용

14기 서울18반 이재윤

소프트맥스(Softmax) 함수

- 모델의 출력(점수)들을 받아서, 모든 클래스에 대한 확률의 총합이 반드시 1이 되도록 만들어주는 함수
- 즉, 전체를 하나의 확률 분포로 바꿈
- 예를 들어 입력 데이터가 [개: 2.0, 고양이: 1.0, 새: 0.1] 일 때,
  - 소프트맥스 함수의 출력: [개: 0.7, 고양이: 0.2, 새: 0.1] (총합 = 1.0)

$$p_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_j}}$$

$$j = 1, 2, \ldots, K$$

- 여러 선택지 중 단 하나의 정답을 고르는 "다중 클래스 분류"에 사용 됨

❖ 헷갈릴 수 있는 시그모이드(Sigmoid) 와의 차이

- 문제: 이 사진이 '고양이'일 확률은? (0.7) → "Yes"
- 문제: 이 사진이 '새'일 확률은? (0.1) → "No"
- 독립적인 출력을 하며, 모든 출력값의 합이 1이 된다는 보장이 없음
- 이진 분류에 적합

14기 서울18반 이재윤

소프트맥스 회귀(Softmax Regression)

- 세 개 이상의 그룹(클래스)에서 하나를 예측(다중 분류)
- 선형 회귀 + 소프트맥스(Softmax) 함수 + 범주형 교차 엔트로피(CCE, Categorical Cross Entropy)

- 범주형 교차 엔트로피 (별로 안중요함)
  - 손실은 실제 정답인 클래스에 대해 모델이 예측한 확률값에만 집중하여 계산
  - 실제 정답(y i)은 정답 클래스에서만 1이고 나머지에서는 0

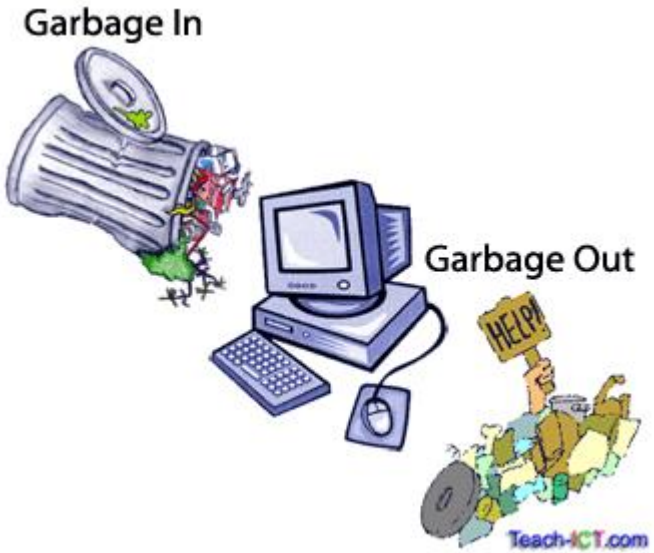
$$\text{Loss} = -\sum_{i=1}^{K} y_i \cdot \log(\hat{p}_i)$$

데이터 전처리

데이터 전처리

- 쓰레기를 넣으면, 쓰레기가 나온다! (Garbage in, Garbage Out)
- 모델이 잘 학습할 수 있도록 데이터를 깨끗하고 정돈된 상태로 바꾸는 모든 과정
- 컴퓨터는 유연하지 못하다..!

1. **결측치**: 비어있는 값(NaN)은 수학 계산을 불가능하게 만듦
2. **문자열/범주형 데이터**: '사과', '바나나' 같은 문자열을 직접 계산할 수 없음
3. **스케일링**: '키(cm)', '몸무게(kg)'처럼 단위와 범위가 전혀 다른 데이터가 섞이면, 컴퓨터는 단순하게 "키"가 더 중요한 특성이라고 착각할 수 있음



➢ 위 문제를 해결해보자!

https://suminii.tistory.com/entry/Garbage-In-Garbage-Out-%EB%8B%B9%EC%8B%A0%EC%9D%98-%EB%8D%B0%EC%9D%B4%ED%84%B0-%EB%AF%BF%EC%9D%84%EB%A7%8C%ED%95%9C%EA%B0%80%EC%9A%94

데이터 전처리 - **결측치**

- 비어있는 값으로써, 데이터 전처리에서 일반적으로 가장 먼저 마주하는 문제
- 결측치 처리 방법1) 삭제 (Deletion)
  - 비어있는 데이터가 포함된 행이나 열을 그냥 삭제
  - 간단하고 빠르지만, 소중한 데이터를 잃어버릴 수 있음 (데이터가 적을 때는 치명적)
- 결측치 처리 방법2) 대치(Imputation)
  - 비어있는 값을 그럴듯한 값으로 채워넣기
  - **숫자형 데이터**인 경우 (키, 몸무게, … 등)
    - 평균
    - 중앙값
  - **범주형 데이터** (혈액형, 등급, … 등)
    - 최빈값

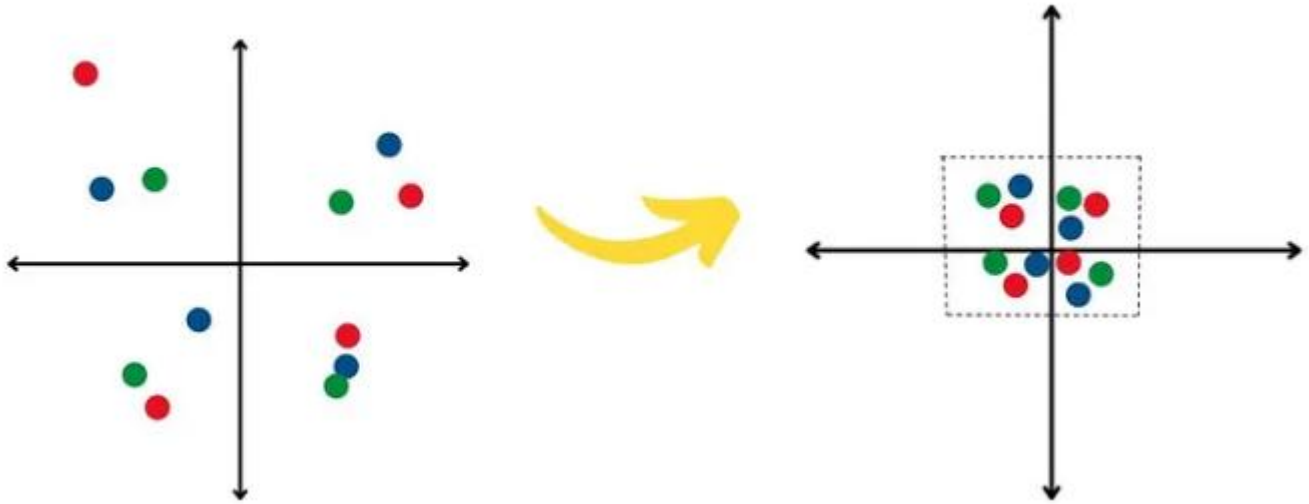
데이터 전처리 - **문자열/범주형 데이터**

- 컴퓨터가 알아들을 수 있도록 문자 데이터를 숫자로 변환하는 과정
- 레이블 인코딩 / 원-핫 인코딩 / 순서형 인코딩 / 임베딩 등 다양한 방식으로 변환 (*이후에 학습 예정)

https://ablearn.kr/newsletter/?bmode=view&idx=13435651

데이터 전처리 - 스케일링

- 각 특성의 단위를 맞춰서 모델이 공평하게 학습하도록 만드는 과정
- 마치 키(180cm)와 몸무게(70kg)를 비슷한 범위의 값으로 조정
- 대표적인 방법으로 표준화와 정규화가 있음



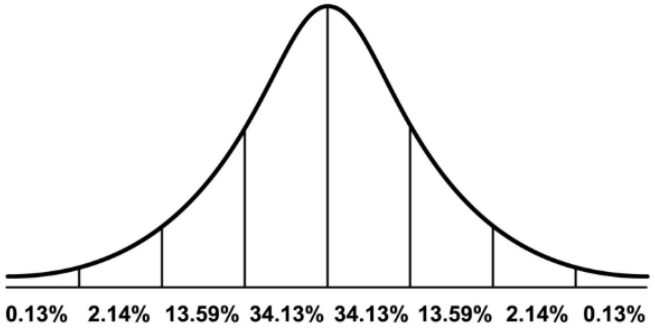
스케일링 전

스케일링 후

https://python.plainenglish.io/understanding-scaling-and-its-common-techniques-064207cf0571

데이터 전처리 - 스케일링 - **표준화**

- 모델에게 '나이' 와 '연봉' 데이터를 그대로 주면 어떻게 될까?
  - ➢ 값의 범위가 훨씬 큰 '연봉'을 더 중요한 정보로 착각할 수 있음
- 제각각인 데이터의 단위를 공평하게 맞춰주는 작업
- 데이터의 평균을 0, 표준편차를 1로 변환
  - 평균 0 : 모든 데이터를 0을 기준으로 맞춤
  - 표준편차 1 : 각 데이터가 평균으로부터 얼마나 떨어져 있는지(표준편차)를 기준으로 데이터의 스케일을 조절
- 특징
  - 이상치에 상대적으로 덜 민감하고, 정규분포를 따를 때 효과적
    (*__이상치__: 다른 값들에 비해 극단적으로 다른 값)
  - 모든 특성에 공평한 영향력을 부여
  - 경사 하강법, SVM, 거리 기반 알고리즘의 성능 향상



데이터 전처리

- 데이터 전처리 - 스케일링 - **정규화**
  - 흩어져 있는 모든 데이터의 값을 0과 1 사이의 범위로 변환
  - 데이터의 범위를 0 ~ 1 로 명확하게 제한하고 싶은 경우에 활용
    - ex) 이미지 처리에서 픽셀값은 보통 0 ~ 255 로 표현되며, 이를 정규화하면 학습이 훨씬 쉬워짐
  - 특징
    - 데이터의 최소/최대값을 명확히 알면 유용함
    - 이상치에 민감
  - **일반적으로 표준화를 더 선호**

- 데이터 분리 및 파이프라인 구축은 이후에 진행

[참고] 시각화 라이브러리

- **Matplotlib**
  - 가장 기본적이고 강력한 데이터 시각화 라이브러리
  - 높은 자유도를 가지고 있지만, 약간 투박해 보일 수 있음
  - numpy에 최적화된 편
- **Seaborn**
  - Matplotlib을 기반으로 한 시각화 라이브러리
  - 쉬운 문법과 멋진 디자인
  - Pandas에 특화

- 오늘 배울 실습에서 사용되는 Matplotlib, Seaborn 코드는 일단 실행하고, 자세한 내용은 내일 학습

다음 시간에
만나요!