기초 이론부터 실무 실습까지

머신 러닝 익히기

programmers

Part 04. 선형 회귀와 선형 분류

정 정 민

programmers

Chapter 09. 선형 회귀와 선형 분류

1. 선형의 의미
2. 선형 회귀
3. 선형 분류

선형의 의미

선형 관계란

- 과자(가격 1,500원)와 우유(1,200원)를 사기위해 마트에서 장을 본다고 해볼까요?
- 전체 구매 비용($TotalCost$)은 아래와 같은 관계를 만족

$$TotalCost = num_{snack} \times 1500 + num_{milk} \times 1200$$

- [조건] 물건 가격 할인과 서로 다른 물건끼리의 가격 영향은 없음

- 전체 비용은 구매하는 과자와 우유의 수(num)에 영향을 받음
  - 과자 수의 증가는 전체 비용에 영향을 미침
  - 즉, 과자 수의 변화가 특정한 비율로 전체 비용에 영향을 미치고 있음
  - 우유의 수도 마찬가지

- 이처럼 독립 변수(num_{snack}, num_{milk})가 파라미터(1,500원, 1,200원) 값 만큼 일정한 비율로 결과 종속 변수($TotalCost$)에 영향을 미치는 관계를 선형 관계라 함

선형 결합과 선형 모델

- 앞서 본 방식처럼 파라미터들이 어떠한 실수(혹은 벡터)와 가중 합(곱하기 & 더하기)으로 표현된 것을 선형 결합이라고 함
- 선형 결합을 일반적으로 표현하면 아래와 같음

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

- $x_1 \ldots x_n$: 독립 변수 혹은 특징(feature), 보통 입력하는 데이터를 의미
- $w_1 \ldots w_n$: 파라미터, 찾아내야 하는 값

- 파라미터들이 선형 결합을 이루고,
  이것으로 종속 변수의 값을 표현할 수 있을 때
  이것을 선형 모델이라고 함

$$y = w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

- y : 종속 변수

- 이 선형 모델을 그래프로 표현한다면 직선(in 2D) 혹은 평면(in 3D) 혹은 초평면(Hyper-plane, over 4D)이라고 함

선형 의미에서의 흔한 질문

- 선형과 비선형을 구분하는 큰 기준은 종속 변수가 파라미터에 대해 선형적 인지 혹은 비선형적 인지에 따라 다름

$$y = ax_1^2 + bx_1 + cx_2 + d$$

- 위 모델은 선형일까요? 비선형일까요?
- 정답은…

- 관점에 따라 다름!
  - 파라미터 a가 종속 변수에 미치는 영향을 볼 때,
    x_1^2을 x_3라는 새로운 변수로 치환한다면 x_3에 단순 비례한다고 보아 선형 모델로 볼 수 있음
  - 다른 관점으로, 변수 x_1의 입장에서 본다면 이는 비선형 모델로 볼 수 있음

- 너무 기준이 애매한 것 아닌가요??
  - 맞습니다. 하지만 위 식이 정답이라는 것을 알고 보면 '치환하면 되겠다'라고 쉽게 생각 들지만,
  - 식을 찾아가는 모델 서칭 단계에서는 치환을 해야 할지, 지수 연산이 사용될지, exp이 들어갈지 알기 쉽지 않음
  - 따라서, 일단 파라미터가 종속 변수에 미치는 영향이 선형적이라는 '가정'으로 선형 모델을 많이 사용

선형 모델을 학습한다는 것은..

- 종속 변수가 어떠한 값을 갖게 되려면 파라미터의 값이 설정이 되어야 함

$$TotalCost = num_{choco} \times 1500 + num_{milk} \times 1200$$

- 초코 과자가 3개, 우유가 2개라면..
- 전체 금액은 6,900원

- 그런데, 머신러닝 입장에서 모델을 학습 시키는 것은
- 특정한 제약 조건이 주어진 상태에서 파라미터의 적절한 값을 찾는 것!
  - 그 상태란, 보통 Loss를 줄이는 최적의 상태
  - 혹은 성능이 제일 높아지는 상태
  - 등등
- 물론 위 식에서는 편의상 파라미터를 1,500과 1,200으로 고정해 둠!

- 우리가 적절한 값을 찾아야 하는 변수라는 의미로 파라미터(parameter)라고 부름
- 그러한 파라미터는 여러 데이터를 바탕으로 최적의 값을 찾아가야 함

선형 모델의 가정

- 이 선형 모델은 매우 중요한 가정이 있음

- 독립 변수가 “독립” 변수라는 이름을 갖듯 서로 다른 독립 변수는 서로 상관성이 없어야 함

- 만약 두 독립 변수 사이에 높은 상관관계(correlation)이 있다면,
- 다중공선성(multicollinearity)이라는 문제를 일으키게 됨 (이후에 자세히 다룸)

- 이렇게 된다면 정확도와 신뢰성에 저하가 일어남!
- 또한, 해석력에도 복잡성이 증가

- 어느 정도의 상관관계가 있을 순 있지만,
- 이 정도가 너무 크다면 추후에 배우게 될 다른 모델을 선택하는 것이 좋은 방법

선형 회귀

강의 실습 코드 : 링크

의미와 모델 표현 식

- 선형 회귀 모델을 사용한다는 것은
  - 입력 데이터 특징 사이의 독립성을 가정하고
  - 데이터 특징에 대한 선형 결합으로 회귀 문제를 풀겠다는 의미

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_p x_p = Xw$$

$$X = \begin{bmatrix} x_{10} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n0} & \cdots & x_{np} \end{bmatrix}$$

- 출력 결과($\hat{y}$)는 예측값에 해당
- w_0은 절편(혹은 편향)에 해당함
  - 목푯값의 평균이 0에 맞춰 있다면, 절편을 추가하지 않을 수 있음
  - 일반적으로 절편을 포함하는 것이 유리
- 가중합으로 표현되는 다항식은 행렬곱(Xw)으로 간단하게 표현할 수 있음

n : 전체 데이터의 수

p : 입력 데이터가 갖고 있는 특성의 수

비용 함수

- 목푯값과 예측값 사이의 계산을 통해 비용 함수를 정의

- 두 값 모두 실수의 범위를 갖으므로
- 두 값 사이의 양적 차이(잔차)의 제곱 평균으로 (평균 제곱 오차, mean squared error, MSE)
- 비용 함수($J(w)$)를 정의

$$MSE = J(w) = \|y - \hat{y}\|_2^2$$

- 이런 비용 함수를 최소화 하는 파라미터(w)들을 찾아야 함

$$\min_w J(w)$$

- 선형 회귀는 잔차의 제곱값이 갖을 수 있는 최소의 파라미터를 찾는 작업이므로
- 최소제곱법(Ordinary Least squares, OLS)이라고도 함

최적화 방법론

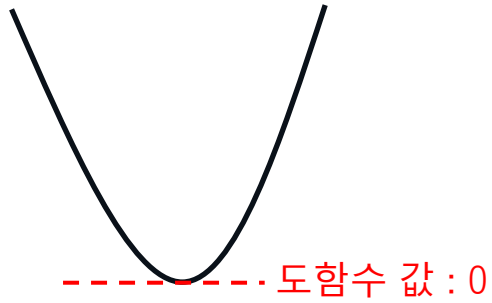
- 최적화란 특정 문제에서 최적의 해를 찾는 과정을 의미함
- '최적'이란 보통 특정 함수의 최소값(min) 혹은 최대값(max) 찾는 것을 가리킴

- 앞서 살펴본 선형 회귀 목적도
- 비용 함수를 최소화하는 파라미터를 찾는 것이므로 최적화 방법론으로 해를 구할 수 있음

- 선형 회귀를 위한 최적화 방법은 정규 방정식 풀이와 경사 하강법이 존재
  - 정규 방정식
    - 파라미터 값을 직접 계산
    - 최소 값을 찾기 위해 도함수가 0이 되는 점을 계산
  - 경사 하강법
    - 비용 함수가 작아지는 방향을 찾아 점진적으로 파라미터를 조정하는 방법

정규 방정식, Normal Equation

- 특정 식이 최소가 되는 지점을 찾는 것은
- 식의 기울기(미분값)가 0이 되는 위치를 찾는 것과 동치



- 따라서, 앞선 비용 함수($J(w)$)의 도함수를 구하고 그것이 0이 되는 파라미터(w)를 구해야 함
- 과정에서 아래 식을 구할 수 있고 이를 정규 방정식이라고 함

$$X^T X w = X^T y$$

- 이때, $X^T X$의 역행렬이 존재한다면, w의 유일한 해를 직접 구할 수 있음

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

- 시간 복잡도 : $O(p^3)$
  - $X^T X$ 의 역행렬을 계산이 필요
  - 일반적으로 $n \gg p$인 경우(입력 데이터 ≫ 특성의 수)에 유용함

$$J(w) = \|y - \hat{y}\|_2^2$$
$$= \|y - Xw\|_2^2$$
$$= (y - Xw)^T (y - Xw)$$
$$= y^T y - 2\mathrm{w^T X^T y} + \mathrm{w^T X^T X w}$$

$$\nabla J(w) = 0 \rightarrow -2X^T y + 2X^T Xw = 0$$
$$\rightarrow X^T Xw = X^T y \text{ (정규 방정식)}$$

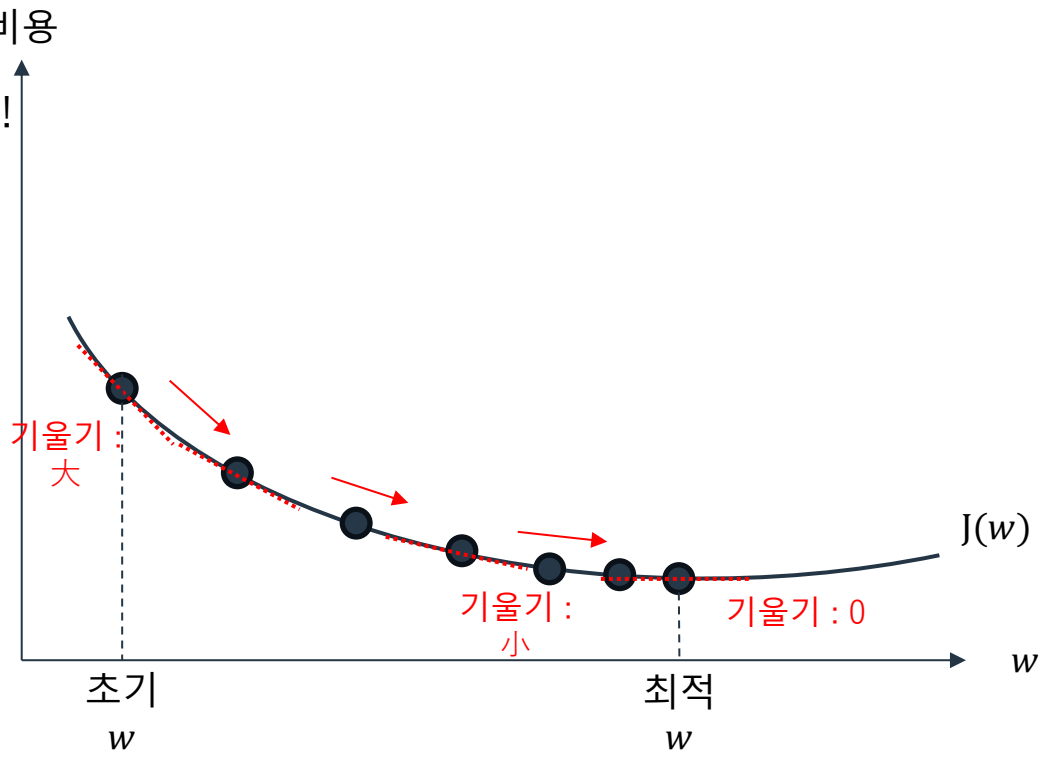
경사 하강법, Gradient Decent

- 비용 함수를 최소화하기 위해 반복해서 파라미터를 조정해가는 방법

- 임의로 잡은 초기 파라미터 값을 기준으로
  비용 함수의 기울기(Gradient)를 계산하여
  기울기가 줄어드는 방향으로 파라미터를 수정 이동

- 반복 수행으로 기울기가 0에 가까워지면(최적값에 도달하면) Stop!

- 주의 사항
  - 적절한 학습률(learning rate, 학습 속도)에 대한 탐구가 필요
    - 값이 너무 작다면 최적화 소요 시간 ↑
    - 값이 너무 크다면 발산 혹은 최적값 도달 가능성 ↓



경사 하강법 – 파라미터 업데이트

- 비용 함수($J(w)$)를 통해 구한 전체 비용을 대상으로
- 각 파라미터의 미분값(편미분)을 구하고
- 현재 값을 기준으로 기울기가 작아지는 방향(-gradient)으로 이동
- 너무 빠른 혹은 느린 학습을 방지하고자 적절한 학습률(learning rate, lr)이 사용

$$J(w) = \|y - \hat{y}\|_2^2 = (y - Xw)^2$$

$$\frac{\partial}{\partial w} J(w) = 2 \cdot X^T \cdot (Xw - y)$$

$$w^{new} = w - lr \cdot \frac{\partial}{\partial w} J(w)$$

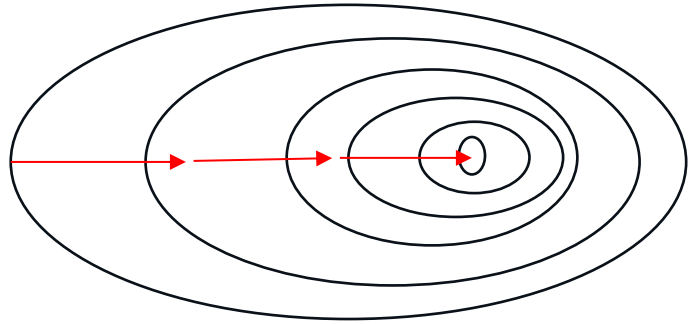
확률적 경사 하강법, Stochastic Gradient Decent

- 경사 하강법은 사용하는 모든 학습 데이터에 대해 기울기 계산이 진행함
- 따라서 학습 데이터가 많은 경우 시간 소요 큼
  - $n \gg p$경우 매우 치명적!

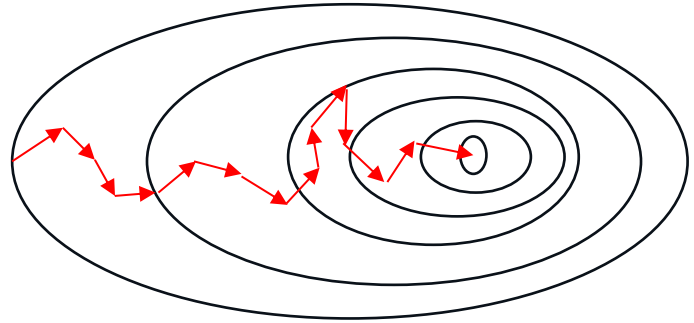
- 이를 극복하기 위해,
  전체 데이터 중 임의로 일부 데이터를 샘플링하여
  그것을 대상으로 경사 하강법을 진행
- 이를 확률적 경사 하강(Stochastic Gradient Decent, SGD) 방법이라고 함

- 작은 데이터로 수정 이동을 반복하므로 빠른 수렴이 가능!

- 데이터가 많거나($n \gg$) 특성 feature가 많은 경우($p \gg$) 사용하기 용이함



Gradient Decent



Stochastic Gradient Decent

뭘 써야 하나요??

- 앞서 살펴본 정규 방정식과 경사 하강법은 모두 선형 회귀를 풀 수 있는 방법
- 방법론은 갖고 있는 데이터 상황에 맞춰 선택할 수 있음

- 정규 방정식은
  - 튜닝할 변수가 없이 명시적으로 해를 제공
  - 하지만 특성의 수가 많다면 계산 복잡도와 메모리가 많이 필요
  - 따라서 훈련 세트가 크지 않은 상황에서 빠르게 해를 구하기 좋음

- 경사 하강법은
  - 특성 수와 샘플의 수에 민감도가 적음
  - 하지만 반복적인 연산이 필요하며 변수 튜닝이 결과에 영향을 줄 수 있음

- 두 방법이 모두 같은 해를 제공하나요?
  - 네, 하지만
  - 선형 모델의 큰 가정인 독립 변수 간의 강한 상관 관계가 있다면, 다중공선성(multicollinearity) 문제가 발생

다중공선성 (multicollinearity)

- 입력 데이터가 갖고 있는 특징값들 사이에 상관 관계가 존재할 때 발생하는 문제 상황
  - 이 상황에서는 머신 러닝 모델이 작은 데이터 변화에도 민감하게 반응
  - 즉, 안정성과 해석력을 저하시킬 수 있음

- 다중공선성이 있는 데이터를 사용할 때,
- 정규 방정식으로 해를 구하는 상황에서는 치명적인 문제가 발생
  - 정규 방정식 해 풀이에 사용되는 $(X^T X)^{-1}$이 존재하지 않을 수 있음
  - $X^T X$ 의 값이 점차 특이 행렬(Singular matrix)에 가까워짐

- 이 상황을 해결하기 위해 SVD-OLS라는 회피 방법이 존재
  - 다중공선성을 해결하기 위한 방법이 아님
  - 강한 다중공선성이 있는 경우 정규 방정식으로 풀 수 없는 해를 다른 방식으로 구하는 방법을 제시

SVD-OLS, Singular Value Decomposition-OLS

- SVD-OLS란 SVD를 활용해 선형 회귀 모델의 해를 구하는 방법
- 학습 데이터를 모아둔 행렬 X에 SVD를 적용해 특이값 분해 $(X = U\Sigma V^T)$

- 특이값이 분해된 입력 X에 OLS 방식의 풀이를 적용

$$J(w) = \|y - \hat{y}\|_2^2 = \|y - Xw\|_2^2 = \|y - U\Sigma V^T w\|_2^2$$

- 변경한 식을 바탕으로 해를 구하면

$$w_{SVD-OLS} = V\Sigma^{-1}U^T y$$

- 시간 복잡도 : $O(np^2)$
  - n : 입력 데이터 수 / p : 사용하는 특성의 수
  - 입력 데이터와 사용하는 특성의 수 모두에 영향을 받음

SVD-OLS VS OLS

- SVD-OLS 방식의 해 : $w_{SVD-OLS} = V\Sigma^{-1}U^T y$
- 단순 OLS 기반의 해 : $w_{OLS} = (X^T X)^{-1} X^T y$

- 주목할 부분은 SVD-OLS에서는 $(X^T X)^{-1}$ 계산 부분이 존재하지 않음
  - 다중공선성이 크게 존재하는 경우
  - $X^T X$ 이 특이 행렬이 되거나 특이 행렬에 가까워짐
  - 그렇다면 역행렬을 계산할 수 없음
- 따라서, $X^T X$의 역행렬을 직접 계산하지 않고도 해를 구할 수 있음
  - 물론 정확한 해를 구하는 것은 아님

- 단, 대신 SVD 계산으로 인한 시간 소요가 더욱 늘어남
  - 단순 OLS : $O(p^3)$
  - SVD-OLS : $O(np^2)$
  - 일반적으로 $n \gg p$ 이므로

- Scikit-learn 패키지 안의 선형 회귀 알고리즘은 SVD류의 방식으로 구현되어 있음

규제(Regulation)를 사용하는 선형 모델

- 과적합 문제 (Over-fitting)
  - 머신 러닝 모델이 피해야 하는 중요한 문제
  - 모델이 학습 데이터에 너무 집중해 일반화가 떨어지는 상황

- 과적합이 발생하면 파라미터인 w의 값이 매우 커지게 됨
- 따라서, w의 값이 너무 커지지 않도록 규제(regulation)를 가해 과적합 문제를 회피할 수 있음

- 선형 모델에서 규제를 추가해 일반화 성능에 도움이 되도록 설계한 모델이 있음
  - 라쏘 회귀 (Lasso regression)
  - 릿지 회귀 (Ridge regression)

라쏘 회귀와 릿지 회귀

- 머신 러닝 모델의 목적은 비용 함수의 최소화

- 이에 따라 기존 비용 함수에 제어할 변수 항목을 추가하는 방식으로 규제를 진행
- 추가로 규제 강도를 조절할 수 있는 새로운 변수 α를 추가

$$J_{Lasso}(w) = \|y - Xw\|_2^2 + \alpha \sum \|w\|_1$$

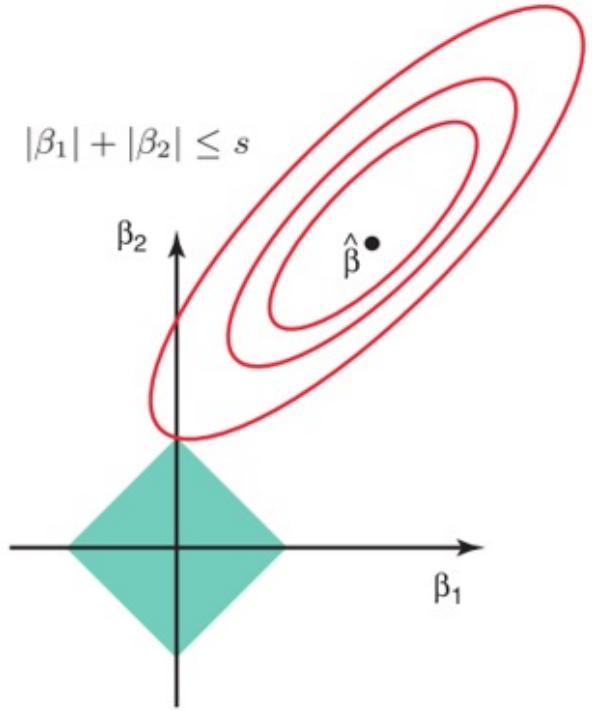
$$J_{Ridge}(w) = \|y - Xw\|_2^2 + \alpha \sum \|w\|_2^2$$

- 라쏘 회귀 비용 함수의 경우 머신 러닝 모델의 파라미터의 절댓값$(l_1\ norm)$의 합 항을 추가
- 릿지의 경우는 제곱합 $(l_2\ norm)$ 항을 추가

라쏘 회귀

$$J_{Lasso}(w) = \|y - Xw\|_2^2 + \alpha \sum \|w\|_1$$

- 라쏘 회귀에 사용되는 L1 규제는 일부 파라미터의 값을 완전히 0으로 만들 수 있음
- 이를 통해 모델이 사용하는 데이터 특성 중 불필요한 특성을 무시하는 효과를 가져올
  - 모델이 단순화되어 해석이 용이함
- 변수가 많고, 일부의 변수가 중요한 역할을 하는 경우 활용될 수 있음!



Lasso Regression

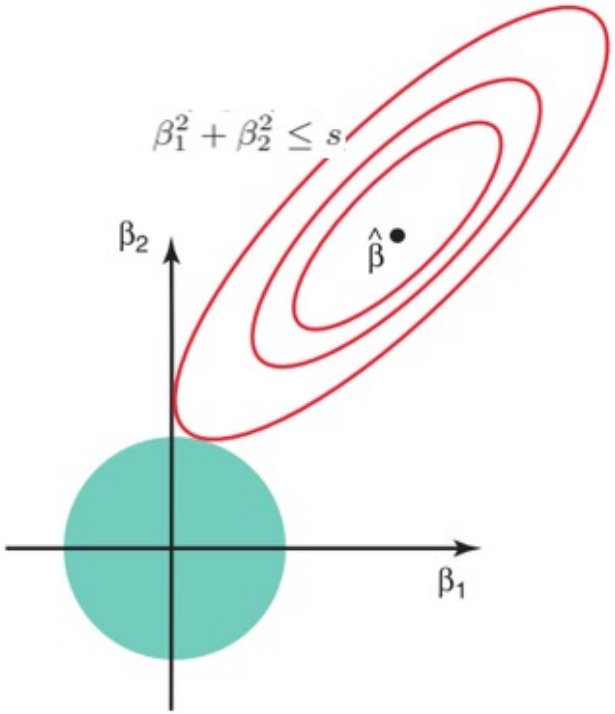
https://www.datacamp.com/tutorial/tutorial-lasso-ridge-regression

릿지 회귀

$$J_{Ridge}(w) = \|y - Xw\|_2^2 + \alpha \sum \|w\|_2^2$$

- 릿지 회귀에 사용되는 L2 규제는 파라미터 값을 적당히 작게 만듦
  - 0에 가까운 값이지만 0이 되지는 않음

- 이는 입력으로 받는 데이터의 모든 부분을 이용해 출력을 판단하는데 사용
- 따라서, 모든 특성이 출력 결과에 적당히 영향을 미치는 경우에 유용함



Ridge Regression

선형 분류

강의 실습 코드 : 링크

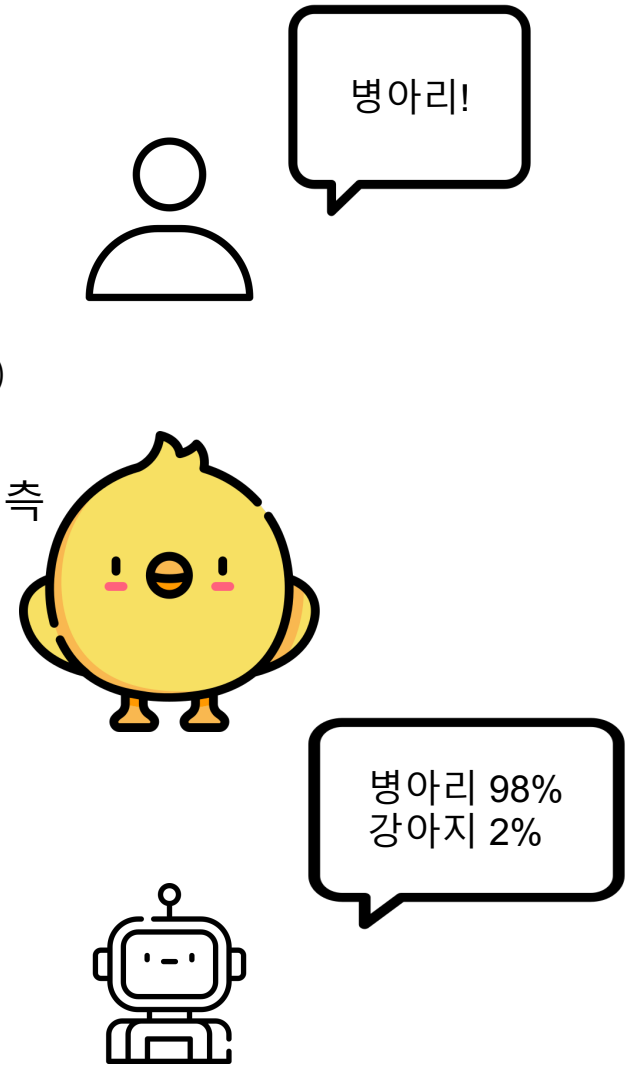
로지스틱 회귀(Logistic Regression) 감 잡기

- 이진 분류 문제를 해결하기 위한 기본 알고리즘 중 하나로
- 입력 데이터가 후보 클래스 중 각각의 클래스일 확률을 예측하는 모델

- 확률이 갖는 값의 범위가 0~1의 실수값이므로
- 그 확률을 직접적으로 예측(→ 확률 추정!)하는 방식으로 문제를 해결
  - 즉, 분류 문제를 풀지만 회귀 방식으로 문제를 접근 (그래서 이름도 로지스틱 '회귀')

- 예측한 특정 클래스의 확률 값이 (일반적으로) 50% 이상이면 해당 클래스에 속한다고 예측
  - 이를 양성(positive) 라고 부르며
  - 그 반대의 경우는 음성(negative)라고 부름

- 다중 클래스 분류 문제의 경우 아래의 접근법이 있음
  - One-vs-One (OvO)
  - One-vs-Rest (OvR)
  - Softmax Regression



로지스틱 회귀

- 로지스틱 회귀도 선형 모델의 조건(독립 변수간 독립성)을 가정

$$\widehat{logit} = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = Xw$$

$$\hat{p} = \sigma\left(\widehat{logit}\right)$$

$$\hat{y} = \begin{cases} 0, & \hat{p} < 0.5 \\ 1, & \hat{p} \geq 0.5 \end{cases}$$

- $\widehat{logit}$: 결과로 뽑아낼 확률값의 로짓(Logit)값
- $\hat{p}$: logit 결과를 확률의 형태로 변경한 확률 추정치
- σ : 로짓(Logit)을 확률로 변경하기 위한 함수 (로지스틱 함수)
- $\hat{y}$: 추정 확률로부터 구한 머신 러닝 모델의 예측

로짓(로그 오즈), 로지스틱 함수

- 오즈(odds)란, 특정 사건이 발생할 확률(p)과 발생하지 않을 확률($1-p$)의 비율

$$odds = \frac{p}{1-p}$$

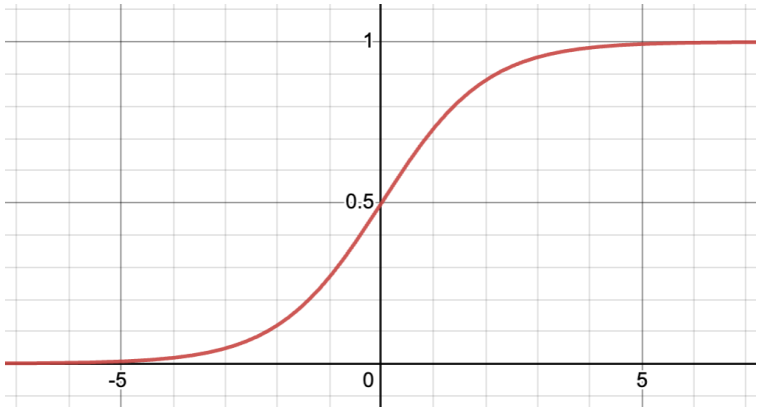
- 로그 오즈(log-odds)란, 오즈(odds)에 log 함수를 씌운 결과로 로짓(logit)이라고도 함
  - 결과를 $\pm\infty$의 범위를 갖도록 변환

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

- 로지스틱(logistic)이란, 로그 오즈의 역함수로 로짓(logit)을 입력으로 주면 확률 p를 반환
  - 따라서, 출력은 항상 0~1사이의 값을 갖고 있음
  - 함수의 개형이 S자 모양이라 Sigmoid(σ) 라고 부름

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

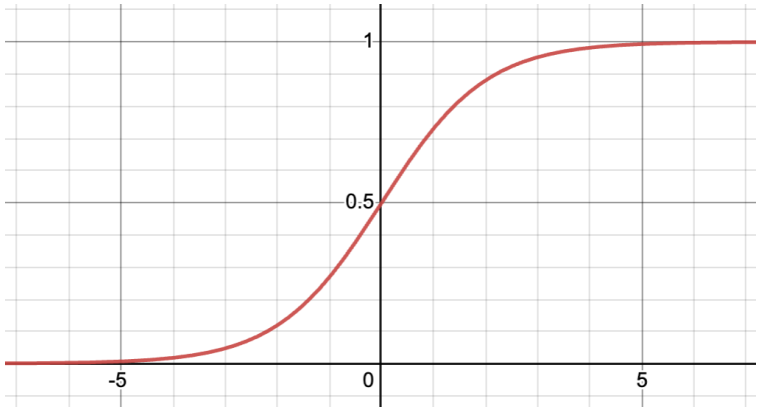
$$\sigma(logit) = p$$



용어는 알겠는데... 그게 어쨌다는거죠??

- 로지스틱 회귀 모델의 출력 결과는 로짓(logit, log-odds)에 해당
- 따라서 특정 클래스일 확률(p)을 알기 위해 로지스틱 함수를 활용

- 왜 확률(p)를 바로 구하지 않는 건가요??
  - 선형 모델은 독립변수의 선형 변환에 따른 종속 변수의 관계를 알아보는 식
  - 하지만 확률의 입장에서 변화량이 구간에 따라 다른 의미를 갖음
    - 예를 들어) 0.49 ↔ 0.50 vs 0.98 ↔ 0.99
  - 즉, 출력 결과의 해석이 선형적이지 않을 수 있음
  - 이를 위해 선형적일 수 있는 다른 표현(여기서는 로짓, logit)을 사용



비용 함수 (1/2)

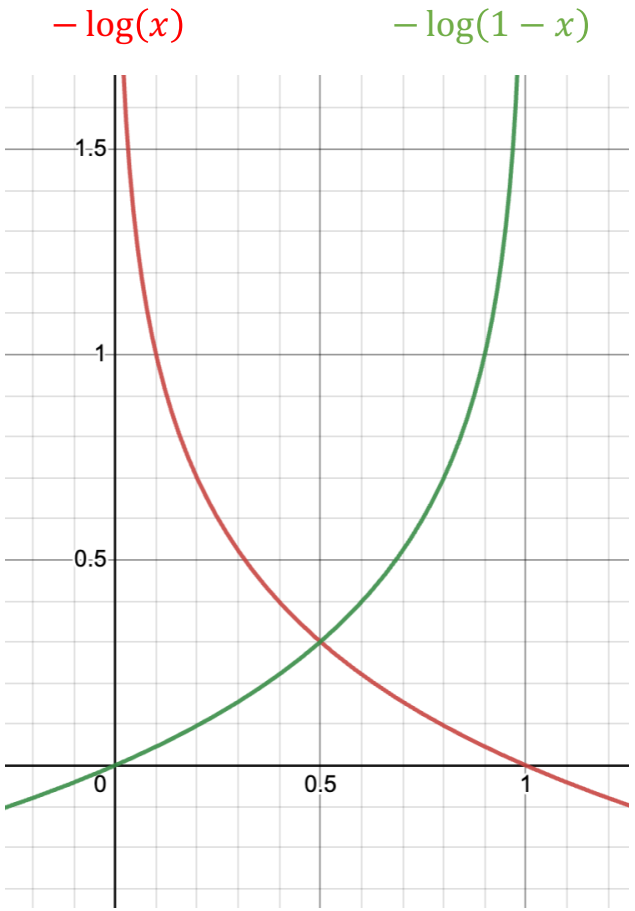
- 분류 문제 역시 정답과의 차이로 비용 함수를 계산

- 분류 문제는 log 함수를 이용한 로그 손실(log loss) 값을 사용
- 단순 회귀 보다는 직관적이지 않은 비용 함수를 사용
- 정답을 잘 예측해야 하는 과정을 담아낼 수 있는 함수를 활용

- 목적 데이터의 클래스가 양성일 경우($y=1$)와 음성일 경우($y=0$)로 나누어
- 하나의 데이터에 대한 비용 함수를 고려하면 아래와 같음

$$J_{oneData}(w) = \begin{cases} -\log(\hat{p}), & y=1\text{인 경우} \\ -\log(1-\hat{p}), & y=0\text{인 경우} \end{cases}$$

- 여기서 $\hat{p}$는 확률 예측값으로 0~1 사이의 값을 갖음
- 위 식에 따르면 $\hat{p}$는 실제 정답 y를 따라가게 됨



비용 함수 (2/2)

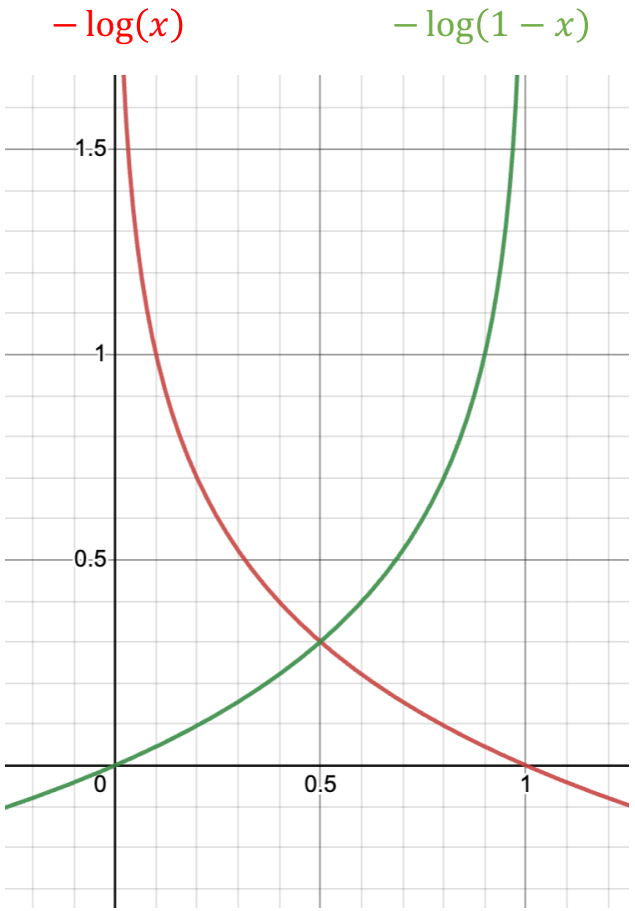
- 앞서 살펴 본 식은 아래와 같이 하나의 식으로 사용할 수 있음

$$J_{oneData}(w) = -(y \cdot \log(\hat{p}) + (1-y) \cdot \log(1-\hat{p}))$$

- 실제 정답(y)의 값에 따라 각 항이 삭제되어 똑같은 식으로 볼 수 있음

- 전체 학습 데이터에 대한 비용 함수는 모든 비용 함수의 평균

$$J(w) = -\frac{1}{N}\sum(y \cdot \log(\hat{p}) + (1-y) \cdot \log(1-\hat{p}))$$



분류 문제의 최적화

$$J(w) = -\frac{1}{N}\sum(y \cdot \log(\hat{p}) + (1-y) \cdot \log(1-\hat{p}))$$

- 분류 문제도 비용 함수를 최소화 시켜야하는 최적화 문제이며 그러한 파라미터(w)들을 찾아야 함

$$\min_{w} J(w)$$

- 분류 문제에서 활용하는 로그 로스(log loss)의 경우,
  회귀 문제의 정규 방정식과 같이 직접적으로 계산 가능한 해가 없음

- 따라서 경사 하강법 방법을 사용해서 파라미터(w)를 찾아야 함

- 또한 이전에 살펴본 라쏘(Lasso) 혹은 릿지(Ridge) 회귀의 규제 방법론을 사용할 수 있음

경사 하강법 – 파라미터 업데이트 (로지스틱 회귀)

- 앞선 선회 회귀와 마찬가지로 분류 문제인 로지스틱 회귀에서도
- 기울기가 작아지는 방향을 찾기 위해
- 각 파라미터의 편미분값을 구해야 함 ($\frac{1}{N}$은 편의상 생략)

$$J(w) = -\sum (y \cdot \log(\hat{p}) + (1-y) \cdot \log(1-\hat{p}))$$

$$\frac{\partial J(w)}{\partial w} = \frac{\partial J(w)}{\partial p} \cdot \frac{\partial p}{\partial logit} \cdot \frac{\partial logit}{\partial w}$$

$$\frac{\partial J(w)}{\partial p} = -\frac{y}{p} + \frac{1-y}{1-p}$$

$$\frac{\partial p}{\partial logit} = p(1-p)$$

$$\frac{\partial logt}{\partial w} = x$$

- Chain rule에 의해 연쇄적으로 계산하면 결과적으로 편미분값은 아래와 같음

$$\frac{\partial J(w)}{\partial w} = (p-y)x$$

- 이를 이용해 새로운 파라미터로 업데이트

$$w^{new} = w - lr \cdot \frac{\partial}{\partial w} J(w)$$

E.O.D