

선형 회귀

삶의 만족도 = $\theta_0 + \theta_1 \times 1인당 GDP$ ($\theta_0 = bias$, $\theta_1 = 자중치$)

$\Rightarrow \hat{y} = h_{\theta}(x) = \theta^T \cdot x$ (θ : 편향, 자중치들, θ^T : 행벡터
 $x: x_0 \sim x_n$ 까지 담은 특성 벡터)

비용함수

$$MSE(x, h_{\theta}) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

정규방정식(Normal Equation)

: 비용 함수를 최소화하는 theta를 찾기 위한 방법

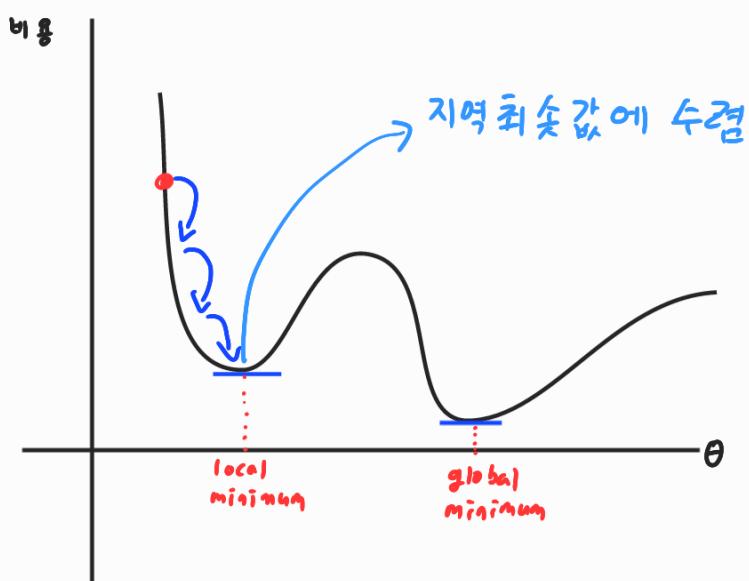
$$\hat{\theta} = (X^T \cdot X)^{-1} \cdot X^T \cdot \underbrace{y}_{\text{타겟 벡터}}$$

경사 하강법(Gradient Descent)

: 파라미터 벡터 theta에 대해 비용 함수의 현재 gradient 계산

-> gradient가 0이 되면 최솟값에 도달한 것

1. θ 를 임의의 값으로 시작 (random initialization)
2. 비용함수(MSE) 가 감소되는 방향으로 진행
3. 이 때 학습률 (learning rate) 조절 \Rightarrow Step
 - 학습률이 작을 때 : 반복 \uparrow \rightarrow 시간 \uparrow
 - 학습률이 높을 때 : 값이 불안
4. 경사 하강법 문제점



MSE 비용 함수의 특징 : 볼록함수(convex function)

-> 지역 최솟값이 없고 하나의 전역 최솟값만 존재

-> 연속된 함수, 기울기가 갑자기 변하지 않음

=> 결국 전역 최솟값으로 수렴할 수 있다!

* 주의 할 점!! => 반드시 모든 특성이 같은 스케일을 갖고 있어야 한다

배치 경사 하강법(Batch Gradient Descent)

- 편도 함수(partial derivative) : 모델 파라미터 theta가 조금 변경될 때 비용 함수가 얼마나 바뀌는지 계산하는 함수

$$\frac{\partial}{\partial \theta_j} MSE(\theta) = \frac{2}{m} \sum_{i=1}^n (\theta^T \cdot x^{(i)} - y^{(i)}) x_j^{(i)}$$

⇒ 비용함수의 편도함수

$$\nabla_{\theta} MSE(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} MSE(\theta) \\ \frac{\partial}{\partial \theta_1} MSE(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} MSE(\theta) \end{pmatrix} = \frac{2}{m} X^T (X \cdot \theta - y)$$

⇒ 비용함수의
그레이디언트 벡터

→ 매 경사하강법 스텝에서 전체 훈련 세트 X에 대해 계산

⇒ 배치 경사하강법

특징 : 1. 매우 느리다. (매우 큰 훈련세트에선)

2. 특정 수에 민감하지 않다.

$$\theta^{\text{next step}} = \theta - \eta \nabla_{\theta} MSE(\theta)$$

⇒ 경사 하강법의 스텝

↳ 학습률 (step의 크기 결정)

반복 횟수 결정 방법 : 반복 횟수 ↑, 벡터의 노름이 허용오차 (ϵ) 보다 작아지면 알고리즘 중지
(Early Stopping)

확률적 경사 하강법

- : 매 스텝에서 딱 한 개의 샘플을 무작위로 선택하고 그 하나의 샘플에 대한 gradient를 계산
- : 확률적(=무작위)이기 때문에 불안정 -> 반복 횟수 높여야 함
- : 배치 경사 하강법보다 전역 최솟값을 찾을 확률이 높다
 - 학습 스케줄(learning rate schedule) : 매 반복에서 학습률을 결정하는 함수
 - : 한 반복에서 m (train set size)번 되풀이 되고 이 반복을 epoch이라고 부름

미니배치 경사 하강법(Mini Batch Gradient Descent)

- : 한 개의 샘플이 아닌 미니배치라 부르는 임의의 작은샘플 세트에 대해 그 라디언트 계산
- : SGD보다 덜 불규칙하게 움직임
- : SGD보다 최솟값에 더 가까이 도달하게 될 것
- : 지역 최솟값에서 빠져나오기는 더 힘들 수 있다.

다항회귀(Polynomial Regression)

- : 2차 다항식으로 새로운 특성 추가 후 선형회귀

학습 곡선

- : 과대적합, 과소적합을 판단하기 위해 학습 곡선을 그림

- 편향/분산 trade off

- bias : 잘못된 가정으로 인한 것. 편향이 큰 모델은 과소적합되기가 쉽다
- variance : 훈련 데이터에 작은 변동에 모델이 과도하게 민감하기 때문 자유도가 높은 모델은 높은 분산을 가져 과대적합 되는 경향 가짐
- irreducible error : 데이터 자체에 있는 노이즈 때문에 발생 -> 노이즈 제거 필요

$$\Rightarrow \text{분산} \uparrow \rightarrow \text{편향} \downarrow$$

규제

: 과대적합을 감소시키는 방법, 다항식의 차수를 감소 시킴

- Ridge Regression (L2 Penalty)

$$J(\theta) = \text{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^n \theta_i^2$$

규제항

α = 모델을 얼마나 규제할지 결정

편향 θ_0 는 규제되지 않는다.

w 를 특성의 가중치 ($\theta_1 \sim \theta_n$) 정의 $\Rightarrow \frac{1}{2} (\|w\|_2)^2$

\Rightarrow 입력 스케일에 민감하기 때문에 스케일 맞춰줘야 함

α 를 증가시킬수록 직선에 가까워짐

(= 분산 \downarrow , 편향 \uparrow)

$$\hat{\theta} = (X^T \cdot X + \alpha I)^{-1} \cdot X^T \cdot y \Rightarrow \text{Ridge 정규 방정식}$$

- Lasso Regression

$$J(\theta) = \text{MSE}(\theta) + \alpha \sum_{i=1}^n |\theta_i|$$

규제항

특징: 덜 중요한 가중치를 완전히 제거하는 힘

\Rightarrow 차수가 높은 다항 특성의 가중치를 0으로

\Rightarrow 자동으로 특성선택을 하고 희소 모델을 만듦

Lasso 비용함수 미분 $X \Rightarrow \theta_i = 0$ 일 때 서브그라디언트 벡터 g 를 사용해 6D에 적용 가능

$$g(\theta, \lambda) = \nabla_{\theta} \text{MSE}(\theta) + \alpha \begin{pmatrix} \text{sign}(\theta_1) \\ \text{sign}(\theta_2) \\ \vdots \\ \text{sign}(\theta_n) \end{pmatrix} \quad \text{where } \text{sign}(\theta_i) = \begin{cases} -1 & (\theta < 0) \\ 0 & (\theta = 0) \\ 1 & (\theta > 0) \end{cases}$$

↳ Lasso subgradient vector

- Elastic Net

: Ridge와 라쏘의 절충 모델 (혼합 비율 r 을 사용해 조절)

$$J(\theta) = \text{MSE}(\theta) + r\alpha \sum_{i=1}^n |\theta_i| + \frac{1-r}{2} \alpha \sum_{i=1}^n \theta_i^2$$

↳ Lasso ↳ Ridge

if $r=1 \Rightarrow \text{Lasso}$
 $r=0 \Rightarrow \text{Ridge}$

- Early Stopping

: Epoch이 진행될수록 validation error가 증가하는 경우 생김 (=과대적합)
 -> 예러가 최소에 도달하는 즉시 훈련을 멈춤

- 로지스틱회귀(Logistic Regression)

: 분류 알고리즘, 샘플이 특정 클래스에 속할 확률 추정

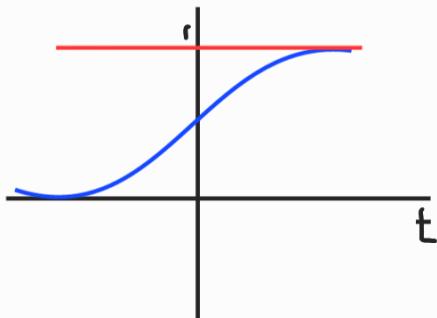
- 확률 추정

: 입력 특성의 가중치 합을 계산. 결과값의 로짓(logit)을 출력

$$\hat{p} = h_{\theta}(x) = \sigma(\theta^T \cdot x) \Rightarrow \text{로지스틱 회귀 모델의 확률 추정 (벡터 표현식)}$$

↳ logit (=sigmoid function)

$$\sigma(t) = \frac{1}{1 + \exp(-t)} \Rightarrow \text{로지스틱 함수}$$



$$\hat{y} = \begin{cases} 0 & \hat{p} < 0.5 \\ 1 & \hat{p} \geq 0.5 \end{cases} \Leftarrow x \text{가 양성일 확률}$$

↳ $\theta^T \cdot x$ 가 양수일 때 1, 음수: 0

- 훈련과 비용함수

: 모델의 파라미터 벡터 theta를 찾기 위한 목적(확률을 추정하기 위해)

$$c(\theta) = \begin{cases} -\log(\hat{p}) & y=1 일 때 \\ -\log(1-\hat{p}) & y=0 일 때 \end{cases}$$

\Rightarrow 하나의 훈련 샘플에 대한 비용함수

$\int_{t \rightarrow 0} -\log(t) = \infty$ 이므로 타당, \therefore if 양성샘플 $y=0$ 이라 하면 비용 \uparrow

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(\hat{p}^{(i)}) + (1-y^{(i)}) \log(1-\hat{p}^{(i)})]$$

\Rightarrow 로지스틱 회귀의 손실함수

θ_j 에 대한 편미분

\downarrow
log Loss : 전체 훈련 세트에 대한 비용함수
(모든 훈련 샘플의 비용을 평균)

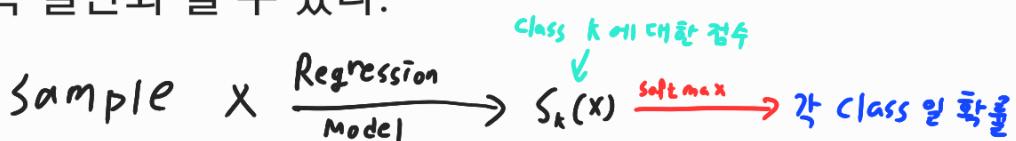
$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m (\sigma(\theta^T \cdot z^{(i)}) - y^{(i)}) z_j^{(i)}$$

\Rightarrow 로지스틱 비용 함수의 편도함수

\hookrightarrow 각 샘플에 대해 예측 오차를 계산하고 J 번째 틱성값을 곱해서 모든 샘플을

- 소프트맥스 회귀

: 여러개의 이진 분류기를 훈련시켜 연결하지 않고 직접 다중 클래스를 지원하도록 일반화 할 수 있다.



$$s_k(x) = (\theta^{(k)})^T \cdot x$$

\Rightarrow class k에 대한 소프트맥스 점수

\hookrightarrow 각 클래스의 파라미터 벡터

\hookrightarrow 파라미터 행렬에 행으로 저장

softmax
function \Rightarrow

$$\hat{P}_k = \sigma(s(x))_k = \frac{\exp(s_k(x))}{\sum_{j=1}^K \exp(s_j(x))}$$

k = class 수, $s(x)$: 샘플 x 에 대한 클래스 점수를 가진 벡터

$\sigma(s(x))_k$ = 샘플 x 에 대한 클래스 점수가 주어졌을 때

샘플이 k 클래스에 속할 확률

$$\hat{y} = \operatorname{argmax} \sigma(s(x))_k \Rightarrow \text{softmax 분류기 예측}$$

Cross Entropy

- : 모델의 타깃 클래스에 대해 높은 확률을 추정하기 위한 목적으로 크로스 엔트로피 비용 함수를 최소화 하는 것 목표
- : 타깃 클래스에 대해 낮은 확률을 예측하는 모델을 억제

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(\hat{p}_k^{(i)}) \Rightarrow \text{Cross Entropy}$$

\Rightarrow i 번째 샘플에 대한 타깃 클래스가 k 일 때 $y_k^{(i)}$ 가 1이고 그 외는 0

$$H(p) = -\sum_x p(x) \log q(x) \Rightarrow \text{두 확률분포 사이의 Cross Entropy}$$

$$\nabla_{\theta^{(k)}} J(\theta) = \frac{1}{m} \sum_{i=1}^m (\hat{p}_k^{(i)} - y_k^{(i)}) x^{(i)} \Rightarrow \text{Cross Entropy의 Gradient Vector}$$