

[7차시 수업]

1)정규화 모델, 주요개념 정리

- 정규화 모델
- 머신러닝,딥러닝 주요개념 정리

2) StyleGAN2

- StyleGAN2 소개
- 실습

3)포터샷 및 베가스

- 포터샷 및 베가스 설치
- 인터페이스 설명

[1교시]

1. 좋은 모델의 조건(정규화 모델)

즉, 정규화모델(Regularization, Ridge Regression)을 뜻함

미래 데이터(testing data)에 대한 예측 성능이 좋은 모델

= 미래 데이터에 대한 expected error가 낮은 모델

$$\begin{aligned}
 \text{Expected MSE} &= E \left[(Y - \hat{Y})^2 | X \right] \\
 &= \sigma^2 + (E[\hat{Y}] - \hat{Y})^2 + E[\hat{Y} - E[\hat{Y}]]^2 \\
 &= \sigma^2 + \text{Bias}^2(\hat{Y}) + \text{Var}(\hat{Y}) \\
 &= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}
 \end{aligned}$$

- Expected MSE를 줄이려면 bias, variance, 혹은 둘다 낮춰야 함
- 그렇지 못하다면 둘 중에 하나라도 작으면 좋음
- Bias가 증가되더라도 variance 감소폭이 더 크다면 expected MSE는 감소 (예측 성능 증가)

estimation

Least squares method (최소제곱법): 평균제곱오차 (MSE)를 최소화하는 회귀계수 $\beta = (\beta_1, \dots, \beta_p)$ 계산

$$\hat{\beta}^{LS} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n (y_i - x_i \beta)^2 \right\} = (X^T X)^{-1} X^T y$$

점추정량
 β : 파라미터

β : 회귀계수

결과값: x와 y의
 데이터의 폴로
 표현

회귀계수 beta에 대한 unbiased estimator 중 가장 분산이 작은 estimator (Best Linear Unbiased Estimator: BLUE, Gauss-Markov Theorem)

점추정량은 unbiased다

1) 점추정량

모수의 추정: 모수의 추정에는 점추정과 구간추정이 있다.

가) 점추정이란? 추출된 표본으로부터 모수의 값에 가까우리라고 예상되는 하나의 값을 제시

나) 구간추정이란? 하나의 값만은 제시하는 것이 아니라 모수를 포함하리라고 예상되는 적절한 구간을 구하는 것.

● 가설검증 방법: 모수(전체 데이터)에 대한 가설 검증을 할 때 추출된 표본 μ 를 기준으로
예) 어느 도시에서 청소년기의 성장에 관한 연구를 하기 위해 중학교 1학년 남학생 30명을 임의 추출하여 키를 측정하였다. 표본으로부터 얻은 30명 키의 평균으로 그 도시 전체 중학교 1학년 남학생의 평균키를 추론할 수 있다.

(1) μ 를 하나의 값으로 추정한다. -> 점추정

(2) μ 를 포함할만한 적당한 구간을 정한다. -> 구간추정

(3) μ 값이 ?년 전의 평균값인 155cm와 다른지 판단한다. -> 가설검정

(1) 점추정(Point Estimation)

점추정이란 추정하고자 하는 하나의 모수에 대하여 모집단에서 임의로 추출된 n 개 표본의 확률변수로 하나의 통계량을 만들고 주어진 표본으로부터 그 값을 계산하여 하나의 수치를 제시하려고 하는 것이다. **이와 같이 모수를 추정하기 위해 만들어진 통계량을 추정량(estimator)이라 하고, 주어진 관측값으로부터 계산된 추정량의 값을 추정치(estimate)라고 한다.**

예를 들어, 위의 예시에서 모평균(해당도시의 중학교 1학년 남학생의 평균키)를 추정한다고 할 때, 직관적으로 가장 타당한 추정량은 표본의 평균

$$\bar{X} = 1/n(X_1 + \dots + X_n)$$

추정량 표본의 평균

이다. 또한, \bar{X} 가 160.2cm라 할 때, 평균키의 추정치는 160.2cm가 된다.

- 추정량은 하나의 확률변수이므로 추출된 표본의 값에 따라 그 값이 달라진다.
- 수치변화를 추정하는 것=> 표준오차(standard error, SE), 표준오차값은 작을수록 좋다.
- **표본평균의 기대 값과 표준오차**는 모집단의 평균(μ)과 표준편차(σ)고 아래와 같이 구할 수 있다.

$$E(\bar{X}) = \mu, S.E.(\bar{X}) = \sigma/\sqrt{n}$$

따라서 \bar{X} 을 가지고 μ 를 추정할 경우 n 이 클수록 표준오차가 작아져 좀 더 정확한 추정이 가능해짐을 알 수 있다.

그러나 $S.E.(\bar{X})$ 를 계산하고자 할 때 σ 이 주어지지 않은 경우가 많은데, 이런 경우에는 σ 를 표본표준편차로 추정하여 사용할 수 있다.

$$s_X = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n - 1}}$$

* 모평균에 대한 점추정

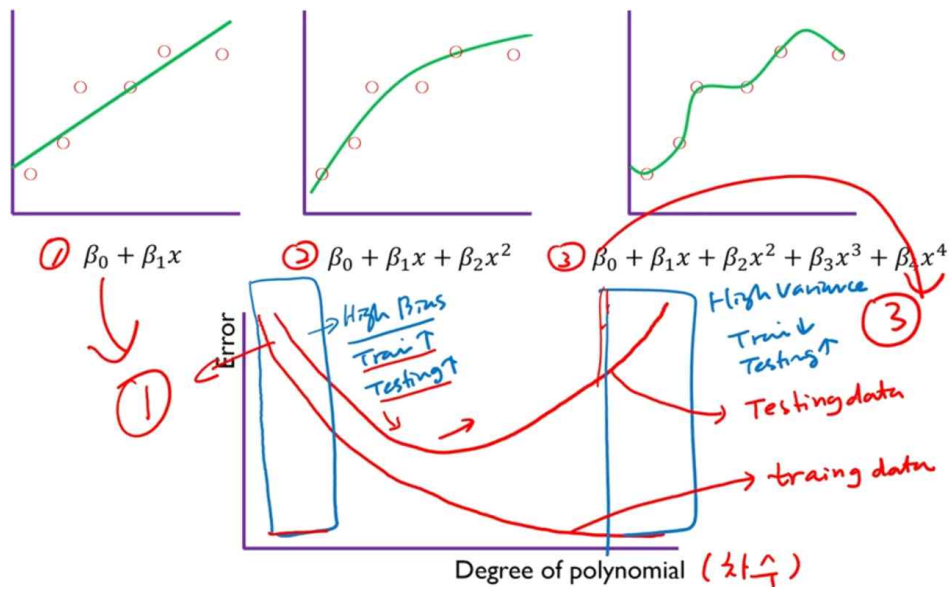
모수 : 모집단의 평균 μ

자료 : 평균이 μ 이고 표준편차가 σ 인 모집단에서 임의추출한 표본 X_1, \dots, X_n

추정량 : 표본평균 \bar{X}

표준오차 : σ/\sqrt{n} 추정된 표준오차 : s/\sqrt{n}

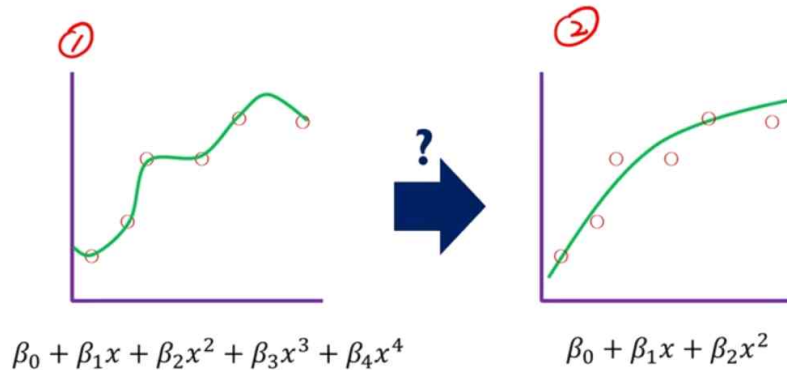
2) Regularization(정규화)



1) 2번이 좋은 모델

2) 1번: Bias가 높다, 3번: Variance가 높다

Regularization(정규화)란?



$$\min_{\beta} \sum_{i=1} (y_i - \hat{y}_i)^2 + 5000\beta_3^2 + 5000\beta_4^2$$

$\beta_3 \approx 0 \quad \beta_4 \approx 0$

β (계수,파라미터)에 제약을 주는것=>정규화(regularization)
 위식은
 β 가 0이 되야 작은 값을 가지게 된다.

$$\beta_1, \beta_2, \dots, \beta_p$$

$$L(\beta) = \min_{\beta} \underbrace{\sum_{i=1} (y_i - \hat{y}_i)^2}_{(1) \text{ Training accuracy}} + \underbrace{\lambda \sum_{j=1}^p \beta_j^2}_{(2) \text{ Generalization accuracy}}$$

λ : regularization parameter that controls the tradeoff between (1) and (2)

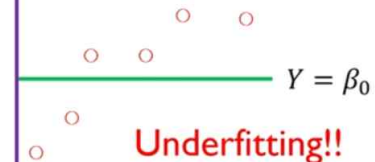
λ (lambda)는 하이퍼파라미터(조절파라미터)

※ 정규화 모델 참조 사이트: <https://www.youtube.com/watch?v=pJCcGK5omhE>

Regularization Concept

$$C(\beta) = \min_{\beta} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

λ very big $\rightarrow \beta_1 \approx 0, \beta_2 \approx 0, \beta_3 \approx 0, \beta_4 \approx 0$



$$\beta_0 + \cancel{\beta_1 x} + \cancel{\beta_2 x^2} + \cancel{\beta_3 x^3} + \cancel{\beta_4 x^4}$$

λ very small \rightarrow



$$\beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$$

λ 값을 너무 높게 주면: β_0 (Y절편)만 남게 된다.==>underfitting

λ 값을 너무 작게 주면: β 값($\beta_1, \beta_2, \dots, \beta_n$)들이 커져서==>overfitting

(1) Regularization Method

- Regularization method는 회귀 계수 beta가 가질 수 있는 값에 제약조건을 부여하는 방법
- 제약조건에 의해 bias가 증가할 수 있지만 variance가 감소함

①

최소제곱법

Least squares method

$$\underset{\beta_1, \beta_2}{\text{minimize}} \sum_{i=1}^n (y_i - x_{i1}\beta_1 - x_{i2}\beta_2)^2$$

②

Regularized method

$$\underset{\beta_1, \beta_2}{\text{minimize}} \sum_{i=1}^n (y_i - x_{i1}\beta_1 - x_{i2}\beta_2)^2$$

$$\text{subject to } \beta_1^2 + \beta_2^2 \leq 30$$

beta 값에 대한 제약 조건

(2) Ridge Regression

L_2 -norm regularization: → 제곱 $L_1 \rightarrow$ 절댓값
 제곱 오차를 최소화하면서 회귀 계수 β 의 L_2 -norm을 제한

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n (y_i - x_i \beta)^2 \quad \text{MSE}$$

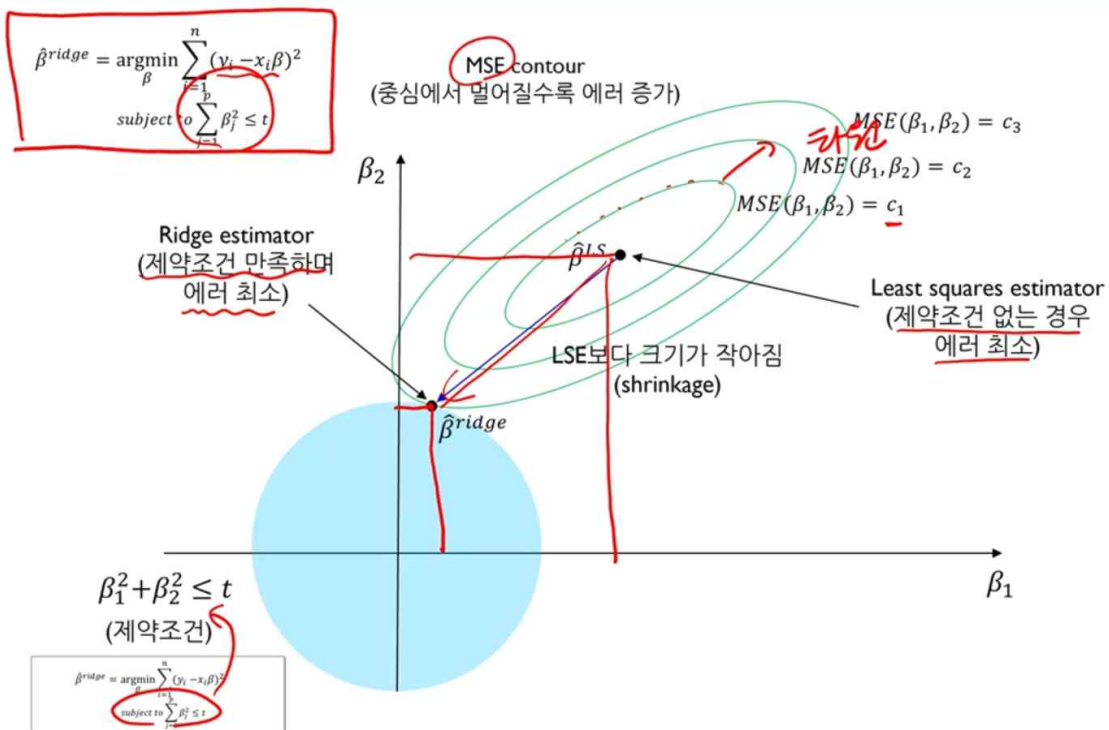
subject to $\sum_{j=1}^p \beta_j^2 \leq t$

\Updownarrow Equivalent (Lagrangian multiplier)

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n (y_i - x_i \beta)^2 + \lambda \sum_{j=1}^p \beta_j^2 \right\}$$

t 와 λ 는 최소값을 만들려는 제약을 주는 기능 동일

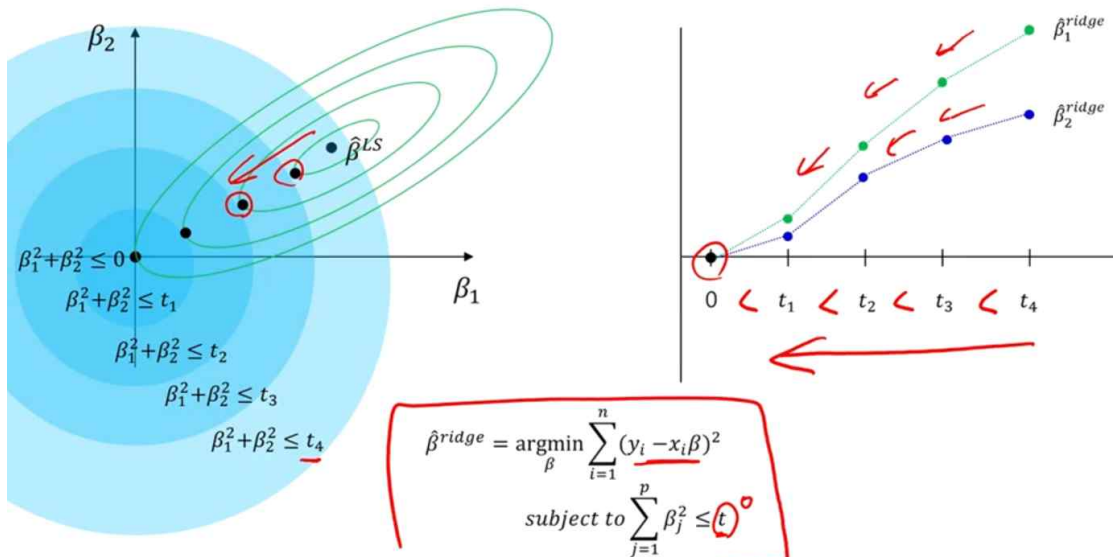
Ridge Regression



β 값을 작게 하는 것(제약조건을 만족하는 최소값) \Rightarrow Ridge Regression

Ridge Solution Path

✓ Solution path: tuning parameter (t) 값에 따른 $\hat{\beta}^{ridge}$ 의 변화



3) LASSO

Least Absolute Shrinkage and Selection Operator

변수 선택 가능

L_1 -norm regularization: 회귀 계수 β 의 L_1 -norm을 제한

✓

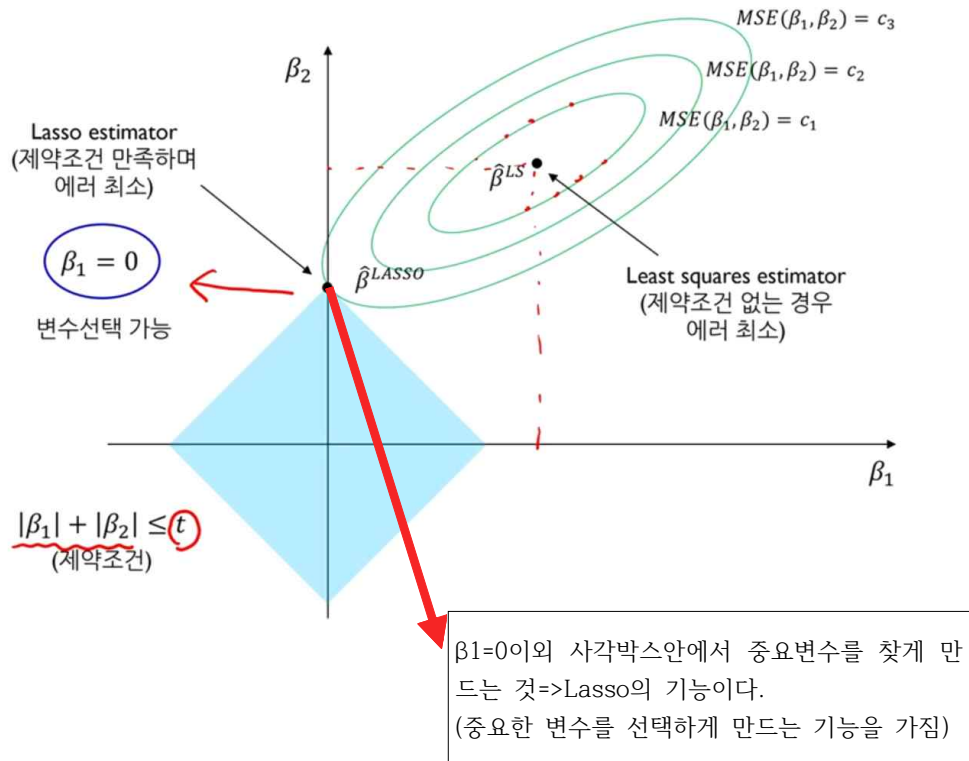
$$\hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n (y_i - x_i \beta)^2$$

subject to $\sum_{j=1}^p |\beta_j| \leq t$ ↓ ↑

⇕ Equivalent

$$\hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n (y_i - x_i \beta)^2 + \lambda \sum_{j=1}^p |\beta_j| \right\}$$

Lasso Solution Path



Lasso Solutions

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n (y_i - x_i \beta)^2 + \lambda \sum_{j=1}^p \beta_j^2 \right\} \Rightarrow \hat{\beta}^{ridge} = (X^T X + \lambda I_p)^{-1} X^T y$$

$$\hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n (y_i - x_i \beta)^2 + \lambda \sum_{j=1}^p |\beta_j| \right\} \Rightarrow \hat{\beta}^{lasso} = ?$$

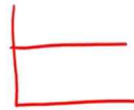
- Ridge와 달리 Lasso formulation은 closed form solution을 구하는 것이 불가능 (L₁-norm 미분 불가능)
- Numerical optimization methods:
 - Quadratic programming techniques (1996, Tibshirani)
 - LARS algorithm (2004, Efron et al.)
 - Coordinate descent algorithm (2007, Friedman et al.)

Lasso Parameter

$$\hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}} \left\{ \underbrace{\sum_{i=1}^n (y_i - x_i \beta)^2}_{MSE} + \lambda \underbrace{\sum_{j=1}^p |\beta_j|}_{\text{penalty}} \right\}$$

$\lambda = 0$: Same as ordinary least square

$\lambda = \infty$: Constant \bar{y}



큰 λ 값



작은 λ 값

적은 변수
간단한 모델
해석 쉬움
높은 학습 오차
(Underfitting 위험 증가)

많은 변수
복잡한 모델
해석 어려움
낮은 학습 오차
(overfitting 위험 증가)

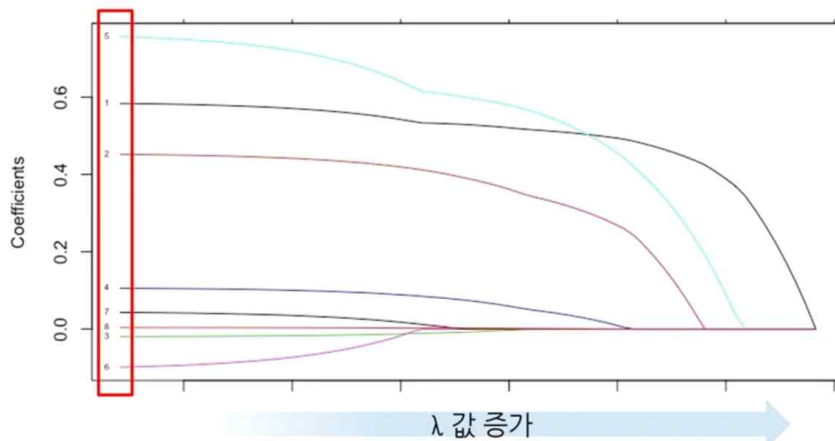
- 람다 값을 점차적으로 증가시킬 때
모든 변수가 최종적으로 0이 된다

Lasso Parameter

$\lambda=0$: 최소제곱법

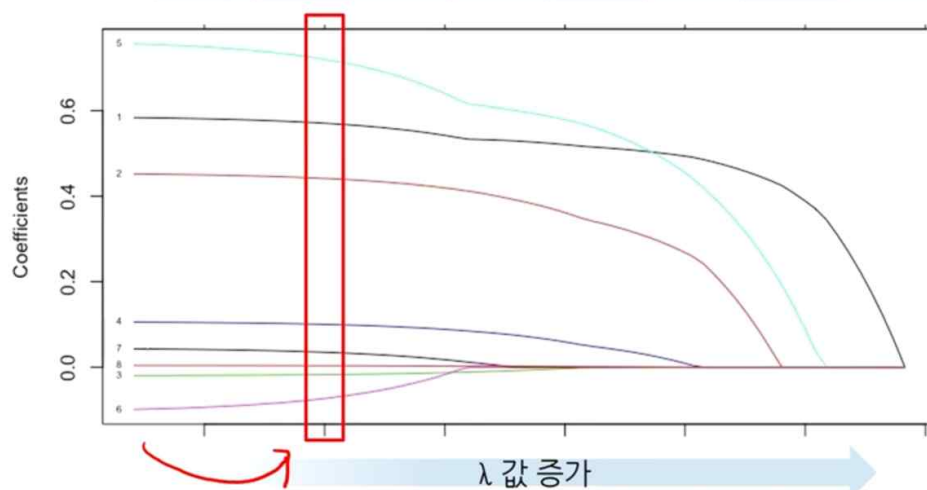
$$\min_{\beta} \left\{ \sum_{i=1}^n \left(y_i - \sum_{j=1}^p x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^p |\beta_j| \right\}$$

β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8
0.584	0.452	-0.019	0.106	0.756	-0.099	0.043	0.004



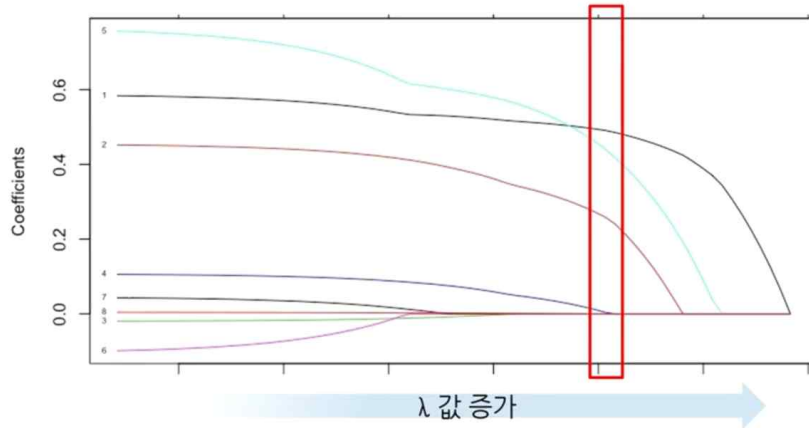
$$\min_{\beta} \left\{ \sum_{i=1}^n \left(y_i - \sum_{j=1}^p x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^p |\beta_j| \right\}$$

β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8
0.580	0.449	-0.018	0.104	0.746	-0.091	0.041	0.004



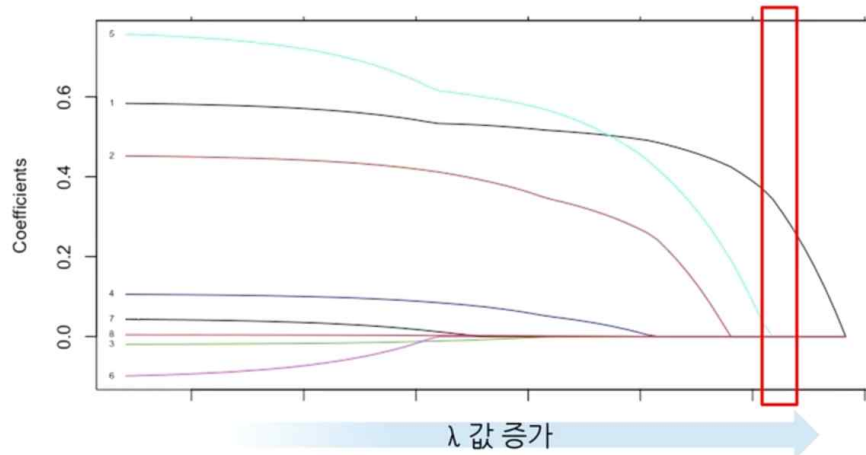
$$\min_{\beta} \left\{ \sum_{i=1}^n \left(y_i - \sum_{j=1}^p x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^p |\beta_j| \right\}$$

β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8
0.486	0.242	0	0	0.419	0	0	0



$$\min_{\beta} \left\{ \sum_{i=1}^n \left(y_i - \sum_{j=1}^p x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^p |\beta_j| \right\}$$

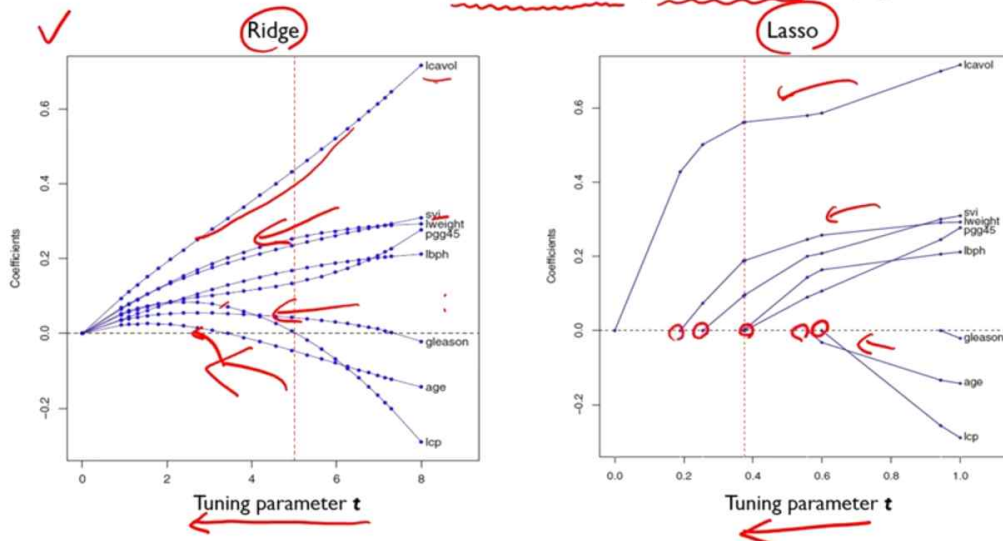
β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8
0.344	0	0	0	0	0	0	0



람다 값을 증가시키면 β_1 (절편)을 제외한 모든 값이 0이 된다.

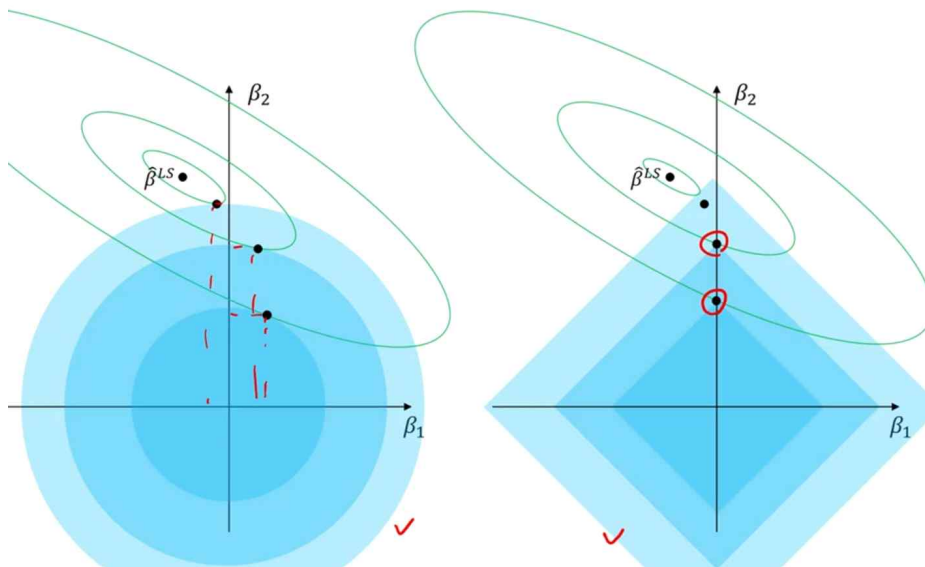
Solution Paths Ridge and Lasso

Prostate cancer data (Y: 전립선 암 항체, X: 환자 의료 데이터)



- Ridge와 Lasso 모두 t 가 작아짐에 따라 모든 계수의 크기가 감소
- Lasso: 예측에 중요하지 않은 변수가 더 빠르게 감소, t 가 작아짐에 따라 예측에 중요하지 않은 변수가 0이 됨

Solution Paths of Ridge and Lasso



Ridge는 β 값이 0이 되는 경우가 없는데 Lasso는 0이 되는 값이 나타남

Ridge vs. Lasso

Ridge	Lasso
L_2 -norm regularization	L_1 -norm regularization
변수 선택 불가능	변수 선택 가능
Closed form solution 존재 (미분으로 구함)	Closed form solution이 존재하지 않음 (numerical optimization 이용)
변수 간 상관관계가 높은 상황 (collinearity)에서 좋은 예측 성능	변수 간 상관관계가 높은 상황에서 ridge에 비해 상대적으로 예측 성능이 떨어짐
크기가 큰 변수를 우선적으로 줄이는 경향이 있음	

Elastic Net

- Elastic net = Ridge + Lasso (L_1 - and L_2 -regularization)
- Elastic net은 상관관계 큰 변수를 동시에 선택/배제하는 특성

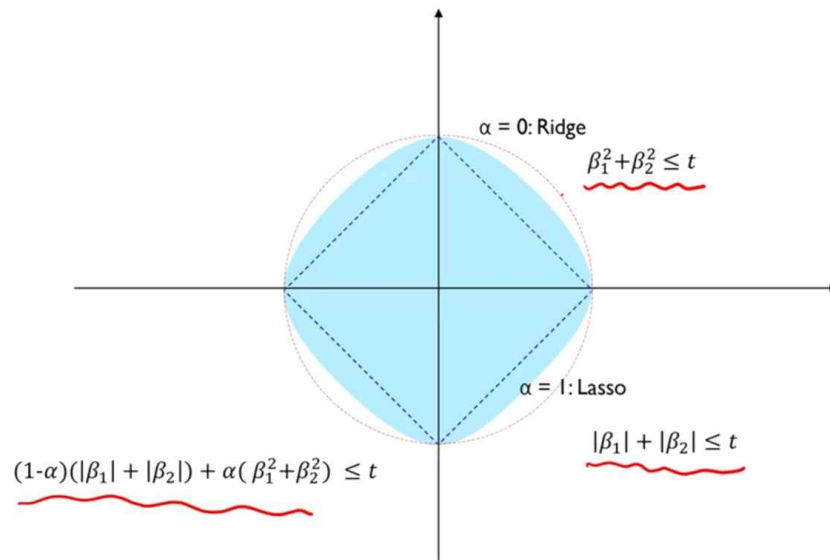
$$\hat{\beta}^{enet} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n (y_i - x_i \beta)^2 \quad \text{MSE}$$

$$\text{subject to } \underbrace{s_1 \sum_{j=1}^p |\beta_j|}_{L_1} + \underbrace{s_1 \sum_{j=1}^p \beta_j^2}_{L_2} \leq t$$

⇕ Equivalent

$$\hat{\beta}^{enet} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n (y_i - x_i \beta)^2 + \underbrace{\lambda_1 \sum_{j=1}^p |\beta_j|}_{L_1} + \underbrace{\lambda_2 \sum_{j=1}^p \beta_j^2}_{L_2} \right\}$$

Elastic Net



Regularization with Prior Knowledge

Prior Knowledge	Regularization Method
상관관계 높은 변수들 동시에 선택	<u>Elastic Net</u>
<u>인접한 변수들 동시에 선택</u>	<u>Fused Lasso</u>
<u>사용자가 정의한 그룹</u> 단위로 변수 선택	<u>Group Lasso</u>
사용자가 정의한 그래프의 연결 관계에 따라 변수 선택	<u>Grace</u>

Fused Lasso

Sparsity
(variable selection,
 L_1 -norm of Lasso)

인접한 변수 beta값 비슷한
크기로 추정
(smoothness)

$$\hat{\beta}^{FL} = \operatorname{argmin}_{\beta} \left\{ \sum_{i=1}^n (y_i - x_i \beta)^2 + \lambda_1 \sum_{j=1}^p |\beta_j| + \lambda_2 \sum_{j=2}^p |\beta_j - \beta_{j-1}| \right\}$$

Sparsity and smoothness via the fused Lasso
(Tibshirani and Saunders, 2005)

Group Lasso

Group Lasso는 사용자가 정의한 그룹 단위로 변수를 선택/배제함

$$\hat{\beta}^{GL} = \operatorname{argmin}_{\beta} \left\{ \sum_{i=1}^n (y_i - x_i \beta)^2 + \lambda \sum_{l=1}^m \sqrt{p_l} \|\beta^{(l)}\|_2 \right\}$$

$p_l =$ Number of variables in group l (group size)

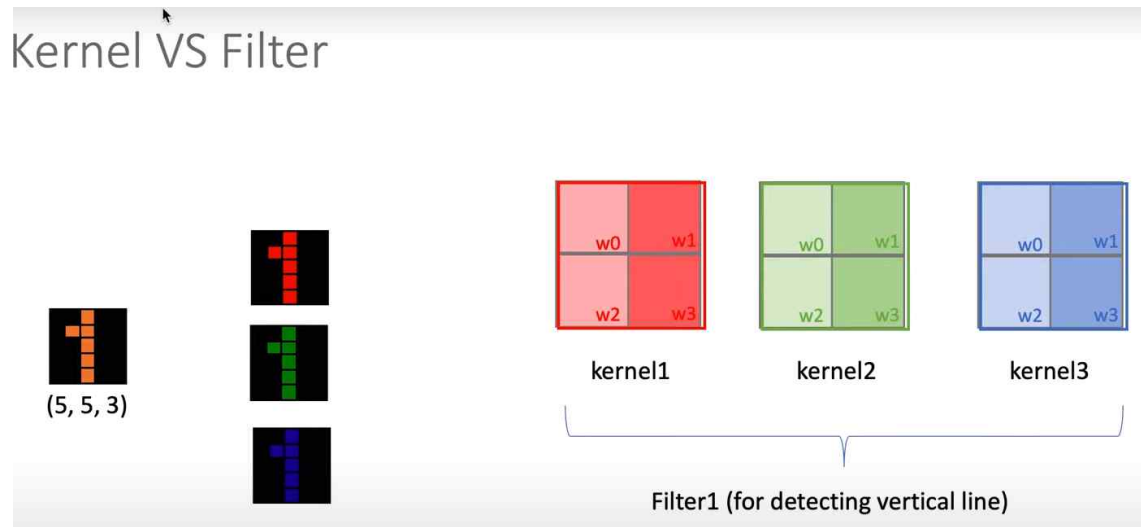
$$\|\beta^{(l)}\|_2 = \left(\sum_{j=1}^{p_l} \beta_{lj}^2 \right)^{\frac{1}{2}}$$

Squared sum of betas in group l

그룹 단위의 변수 선택: Group-wise sparsity

[주요개념 정리]

◆ 필터와 커널의 차이점



Kernel: 2차원 행렬에서 있는 것들

Filter: 하나의 특징을 찾기 위해서 이미지의 개수만큼 커널을 가지고 있다.

참조사이트:



<https://www.youtube.com/watch?v=3exMAKtSGBQ&list=UUxP77kNgVfiiG6CXZ5WMuAQ>

Move filter to detect feature



◆ 평균제곱오차 공식과 코드

RMSLE 평가방법



RMSLE Scorer

Vivek Srinivasan

EDA & Ensemble Model (Top 10 Percentile)

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2}$$

```
def rmsle(y, y_, convertExp=True):
    if convertExp:
        y = np.exp(y),
        y_ = np.exp(y_)
    log1 = np.nan_to_num(np.array([np.log(v + 1) for v in y]))
    log2 = np.nan_to_num(np.array([np.log(v + 1) for v in y_]))
    calc = (log1 - log2) ** 2
    return np.sqrt(np.mean(calc))
```

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2}$$

```
from sklearn.metrics import make_scorer

def rmsle(predicted_values, actual_values):
    # 넘파이로 배열 형태로 바꿔준다.
    predicted_values = np.array(predicted_values)
    actual_values = np.array(actual_values)

    # 예측값과 실제 값에 1을 더하고 로그를 씌워준다.
    log_predict = np.log(predicted_values + 1)
    log_actual = np.log(actual_values + 1)

    # 위에서 계산한 예측값에서 실제값을 빼주고 제곱을 해준다.
    difference = log_predict - log_actual
    # difference = (log_predict - log_actual) ** 2
    difference = np.square(difference)

    # 평균을 낸다.
    mean_difference = difference.mean()

    # 다시 루트를 씌운다.
    score = np.sqrt(mean_difference)

    return score
```

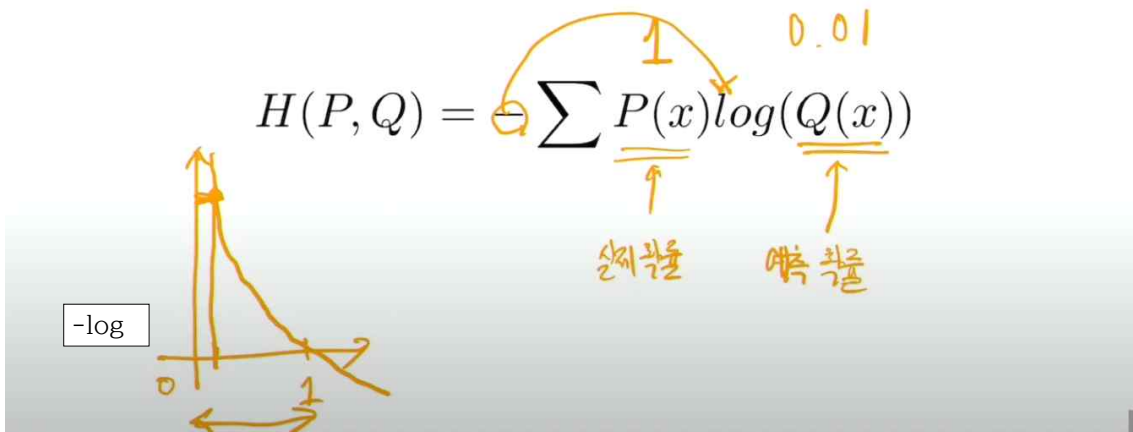
참조사이트:

<https://www.youtube.com/watch?v=95fCw-n5uWM&list=PLaTc2c6yEwmotOgH9PIBpLf3CZEXdD41q&index=2>

■ Cross Entropy

Cross Entropy

Difference between two probability distribution



- cross entropy는 예측을 못할 수 록 커진다.
- 실제 확률(1)에서 예측확률(유저가 예측해보는 확률) 빼기
- 잘 예측하여 예측확률을 0.9인 경우 cross entropy는 작게 된다.

Cost Function

$$cost(W) = \frac{1}{m} \sum_{i=1}^n c(H(x^{(i)}), y^{(i)})$$

$$c(H(x), y) = \begin{cases} -\log(H(x)) & : y = 1 \\ -\log(1 - H(x)) & : y = 0 \end{cases}$$

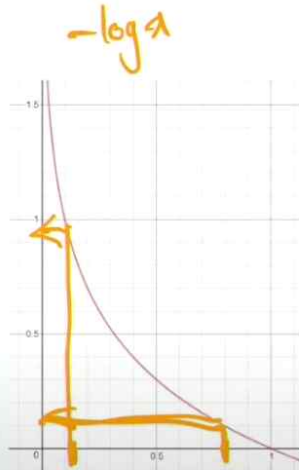
$$c(H(x), y) = -y \log(H(x)) - (1 - y) \log(1 - H(x))$$



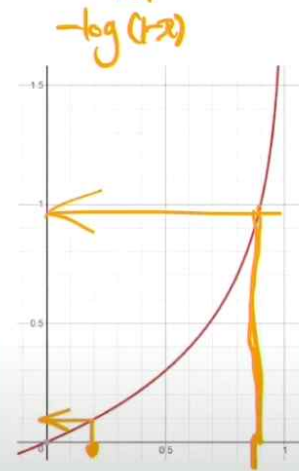
- y=1인 경우: 우측 향이 사라짐
- y=0인 경우: 좌측 향이 사라짐

Cost Function

$$c(H(x), y) = \begin{cases} -\log(H(x)) & : y = 1 \text{ Pass} \\ -\log(1 - H(x)) & : y = 0 \text{ Fail} \end{cases}$$



Pass



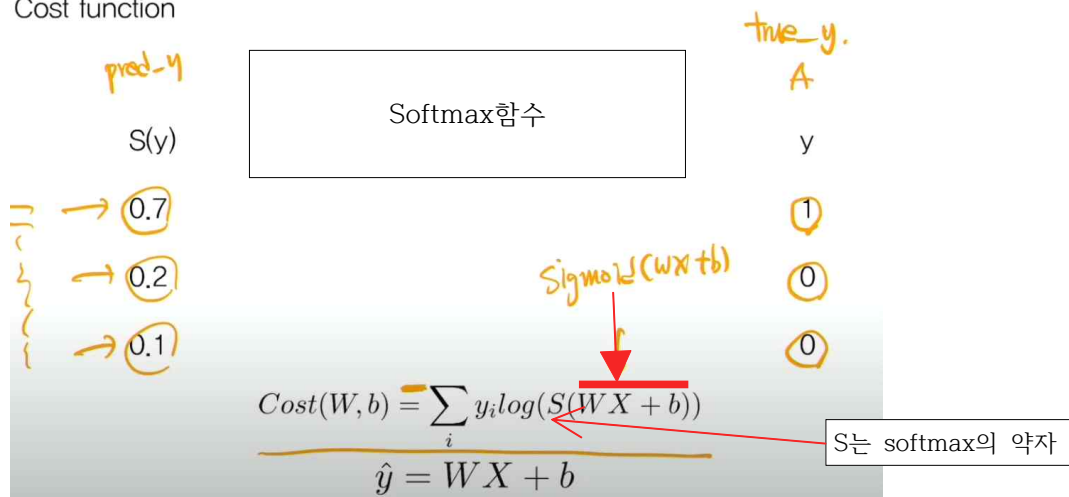
Fail

$$\text{Softmax}(\hat{y}_i) = \frac{e^{\hat{y}_i}}{\sum_j e^{\hat{y}_j}}$$

$$= \frac{e^{y_0}}{e^{y_1} + e^{y_2} + e^{y_3}} = \text{A 1/2 Prob.}$$

분모와 분자의 총합은 =1인 함수=softmax 함수

Cost function



Binary and Multinomial Classification

	Binary	Multinomial
Hypothesis	$Sigmoid(WX)$	$Softmax(WX)$

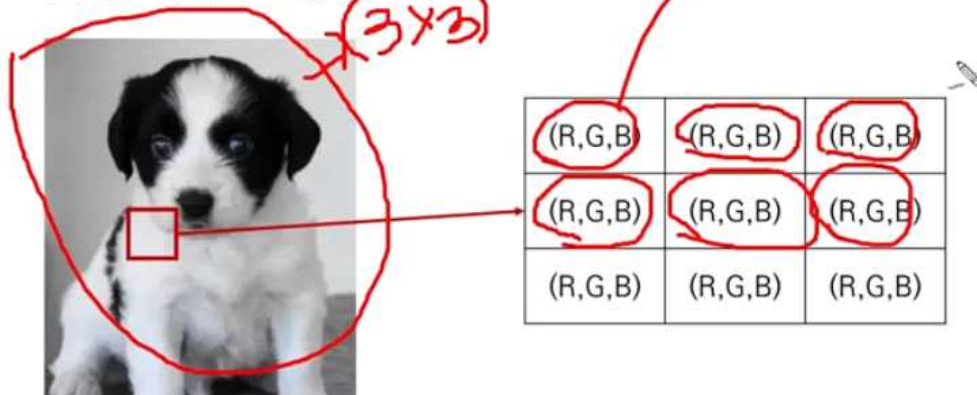
Cost

$$-y \log(H(x)) - (1 - y) \log(1 - H(x)) \quad \sum -y \log(H(X))$$

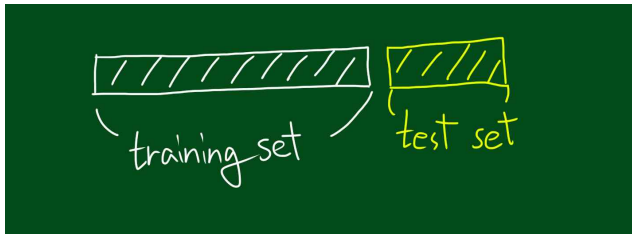
행렬

행렬 사용 예제

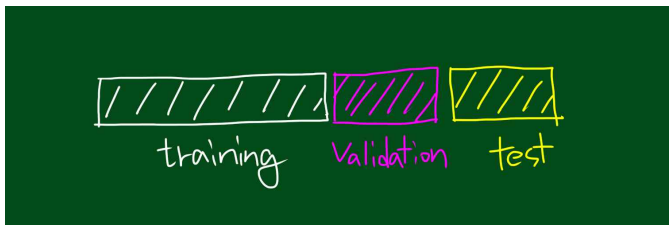
- 이미지는 내부적으로 어떻게 표현될까요?



■ cross validation

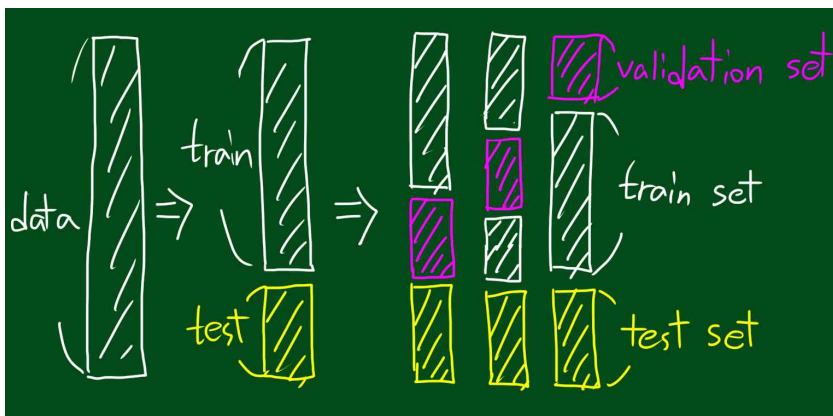


- 데이터셋은 크게 학습데이터와 테스트데이터로 나눈다.
- 문제점: test set이 모형의 parameter 추정에 영향을 미치게 됨.
우리가 원하는 것은 실제 데이터에 성능이 좋은 test set이 되게 하는 것
즉, test set에 적합한 모형을 만들기 보다 실제 데이터에 좋은 성능을 보이는 것을 만드는 것이 목적



그래서 validation set이 필요한 것이다.

- 모형의 parameter 추정에는 validation set을 사용하고,
- 실제세계에 해당하는 데이터가 test set인 것
즉, training data로 학습시키고, validation set으로 parameter 튜닝하고, test set으로 최종 테스트를 하는 것이다.



k-fold cross validation(교차검증)이란?

- training set을 k등분하고 validation set을 여러번 바꿔가며 반복적으로 시행하는 방법
- 위 그림에서는 training set을 3등분했으니 3-fold cross validation에 해당
- 전체 데이터를 k등분한 후 validation set없이 test set 위치를 바꿔가며 테스트 하는 종류도 있음

■ LOSS

1) Cross-Entropy Loss(CE loss)

$$CE = - \sum_i^C t_i \log(s_i)$$

t_i : ground truth (정답)

s_i : 각 클래스 i 에 대한 CNN 마지막 층의 아웃풋인 score 벡터의 i 번째 요소












- (0, 1) 사이 계산 범위를 맞추기 위하여 스코어는 sigmoid activation function과 종종 같이 붙어서 CE loss와 계산된다.
- binary classification 문제에서는 (즉, $C' = 2$), 식을 전개해보면 다음과 같다.

$$CE = - \sum_{i=1}^{C'=2} t_i \log(s_i) = -t_1 \log(s_1) - (1 - t_1) \log(1 - s_1)$$

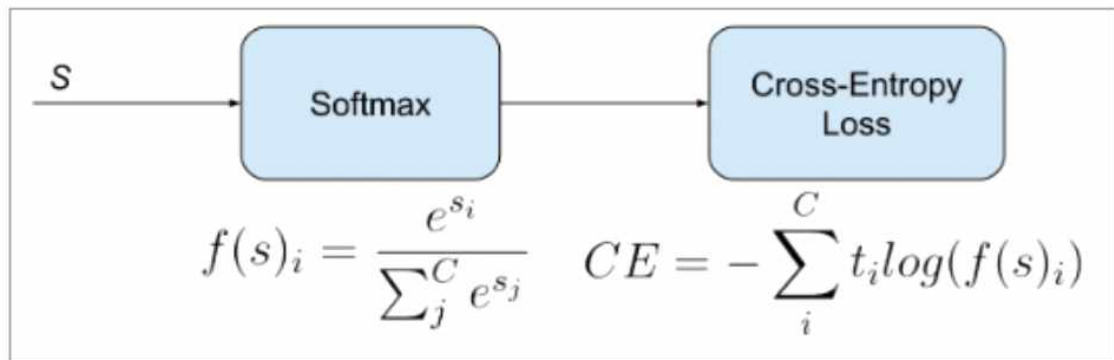
2) Categorical Cross-Entropy Loss(Softmax loss)

- Softmax loss: Softmax activation 뒤에 Cross-Entropy loss를 붙인 형
- Multi-class classification에 사용

※Multi-class classification: 여러 샘플(이미지)에서 하나의 클래스로 분류하는 문제
/ Multi-label Classification은 여러 샘플(이미지)에서 각 샘플마다 여러 클래스가 있을 수 있는 문제

Multi-Class		Multi-Label	
C = 3	Samples	Samples	
			
			
			
	Labels (t)	Labels (t)	
	[0 0 1] [1 0 0] [0 1 0]	[1 0 1] [0 1 0] [1 1 1]	

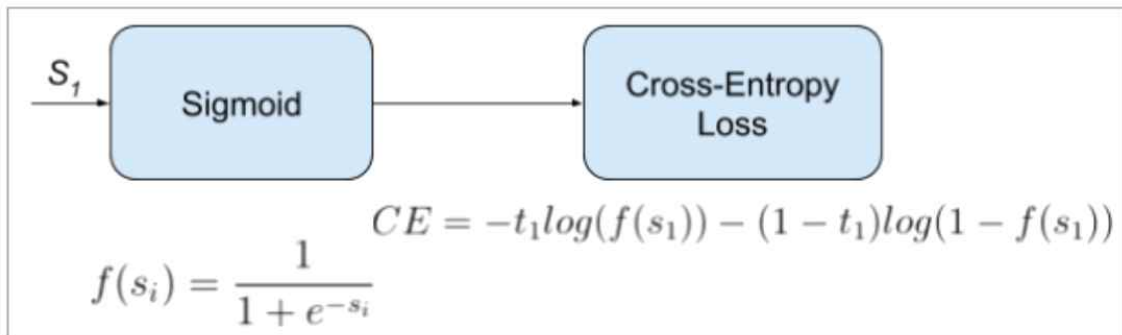
- Softmax loss는
- 분류문제에서 주로 사용하는 활성화함수와 loss이다.
- 분류 문제에서는 MSE(mean square error) loss 보다 CE loss가 더 빨리 수렴한다.
- multi class에서 하나의 클래스를 구분할 때 softmax와 CE loss의 조합을 많이 사용



3) Binary Cross-Entropy Loss

- Sigmoid CE loss: Sigmoid activation 뒤에 Cross-Entropy loss를 붙인 형태
- Multi-label classification에 사용

$$CE = - \sum_{i=1}^{C'=2} t_i \log(f(s_i)) = -t_1 \log(f(s_1)) - (1 - t_1) \log(1 - f(s_1))$$



스칼라: scalar

스칼라는 하나의 숫자만으로 이루어진 데이터를 의미합니다. 스칼라는 보통 x 와 같이 알파벳 소문자로 표기하며 실수(real number)인 숫자 중의 하나이므로 실수 집합 " \mathbb{R} "의 원소라는 의미에서 다음과 같이 표기한다.

$$x \in \mathbb{R}$$

벡터: vector

벡터는 여러 숫자가 순서대로 모여 있는 것으로, 일반적인 일차원 배열이 벡터입니다.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

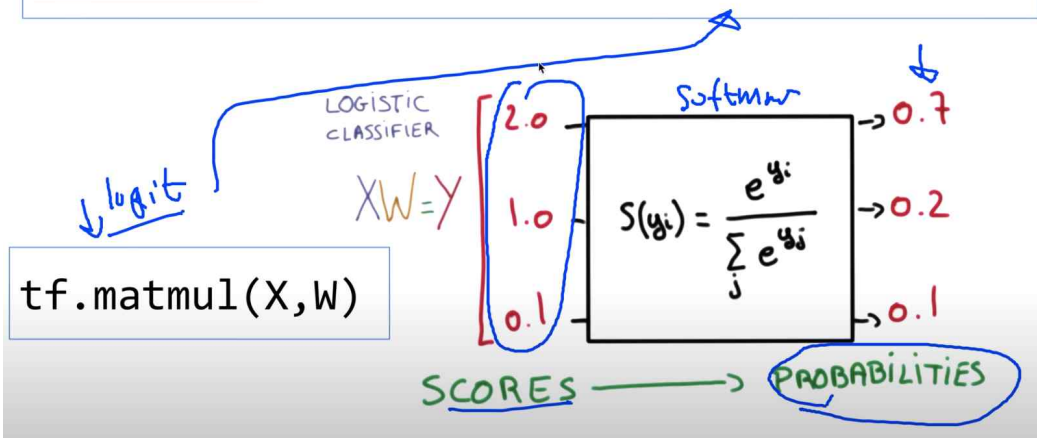
하나의 벡터를 이루는 데이터의 개수를 차원(dimension)이라고 합니다. 위에서 예로든 벡터는 4개의 실수로 이루어져 있고, 4차원 벡터입니다. 이 벡터는 다음과 같이 표기할 수 있습니다.

$$x \in \mathbb{R}^4$$

■ softmax

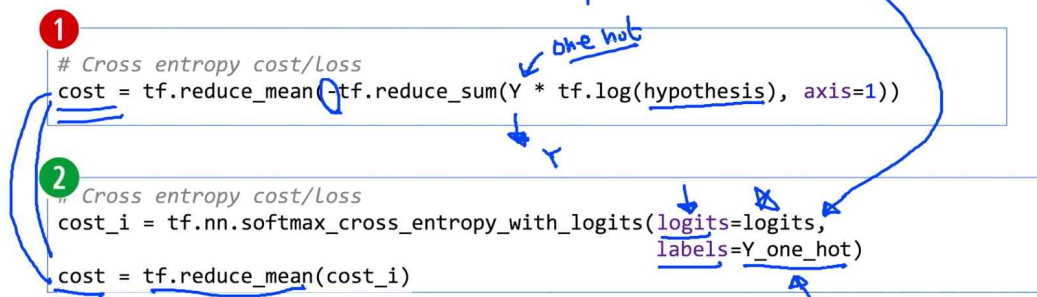
- 텐서플로어 코드

```
hypothesis = tf.nn.softmax(tf.matmul(X,W))
```



softmax_cross_entropy_with_logits

```
logits = tf.matmul(X, W) + b  
hypothesis = tf.nn.softmax(logits)
```



[2교시]

구글콜랩을 통한 학습 StyleGAN2

구글콜랩 코드)

https://colab.research.google.com/drive/1ShgW6wohEFQtqs_znMna3dzrcVoABKIH#scrollTo=Po7eQSxav8qj