모델 선택과 확장

고려대학교 석 준 희

목차

- 모델 선택
- 모델 확장

KOREA UNIVERSITY

모델 선택

학습 모델의 일반적인 표현

- 학습 모델의 파라메터
  - 모델 파라메터: 데이터로부터 학습되는 파라메터로 모델의 일부, 모델이 결정되면 자동으로 결정됨
  - 하이퍼 파라메터 (튜닝 파라메터): 데이터로부터 학습되지 않고 사용자가 직접 지정해주는 파라메터

$$Y = f(X|\theta, \lambda) + \epsilon$$

- 예

방법론	모델 파라메터	하이퍼 파라메터
선형회귀	계수	없음
다항회귀	계수	다항식 차수
KNN	인접한 샘플들	K
트리모델	트리 분할 위치	트리의 크기
인공신경망	각 노드 간의 weight	레이어 수, 노드 개수, 활성화 함수 등

모델 선택 (Model Selection)

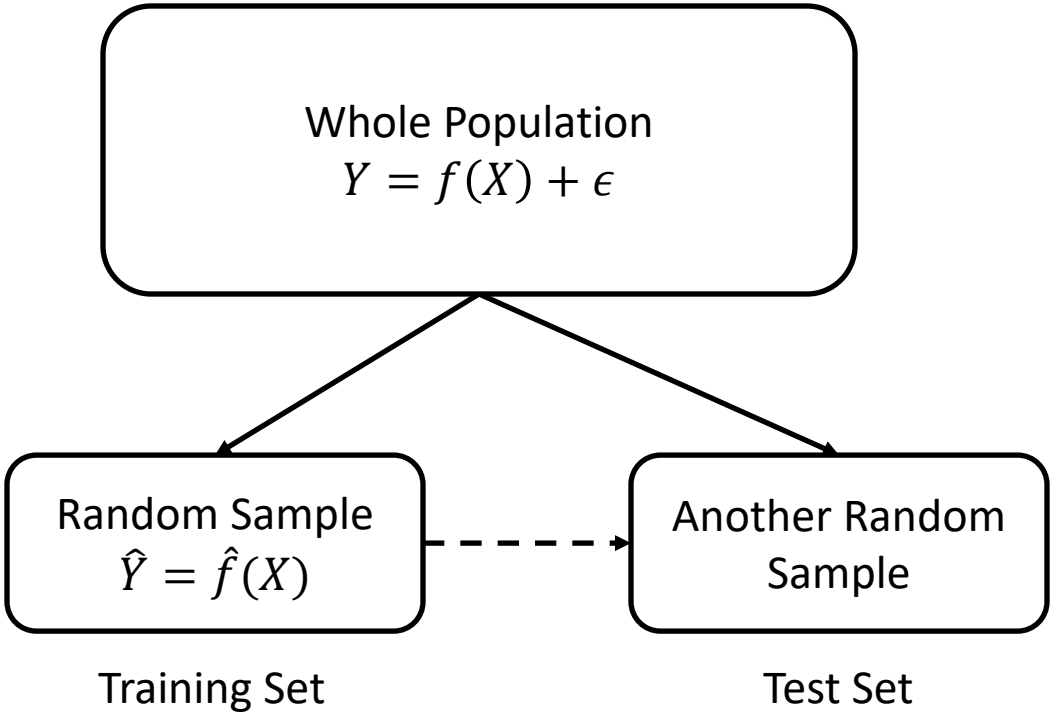
- 많은 모델 중 어느 모델이 가장 좋은 모델일까?
  - $Y = \beta_0 + \beta_1 X$ vs. $Y = \beta_0 + \beta_1 X + \beta_2 X^2$
  - 선형회귀모델 vs. KNN 회귀모델

- 전통적인 통계적 접근
  - 이론적 접근을 통해 결정
  - 모델의 기본이 되는 가정에 대한 확인 (선형성, 정규분포 등)
  - 모델의 적합성(goodness of fit)에 대한 많은 수치들

- 현대 기계학습적 접근
  - 실험적 접근을 통해 결정
  - 훈련-평가 세팅을 시뮬레이션
  - 검증집합(validation set) 혹은 교차검증(cross-validation)을 이용
  - 이러한 방식은 모두 재표집(resampling)에 기반을 둠

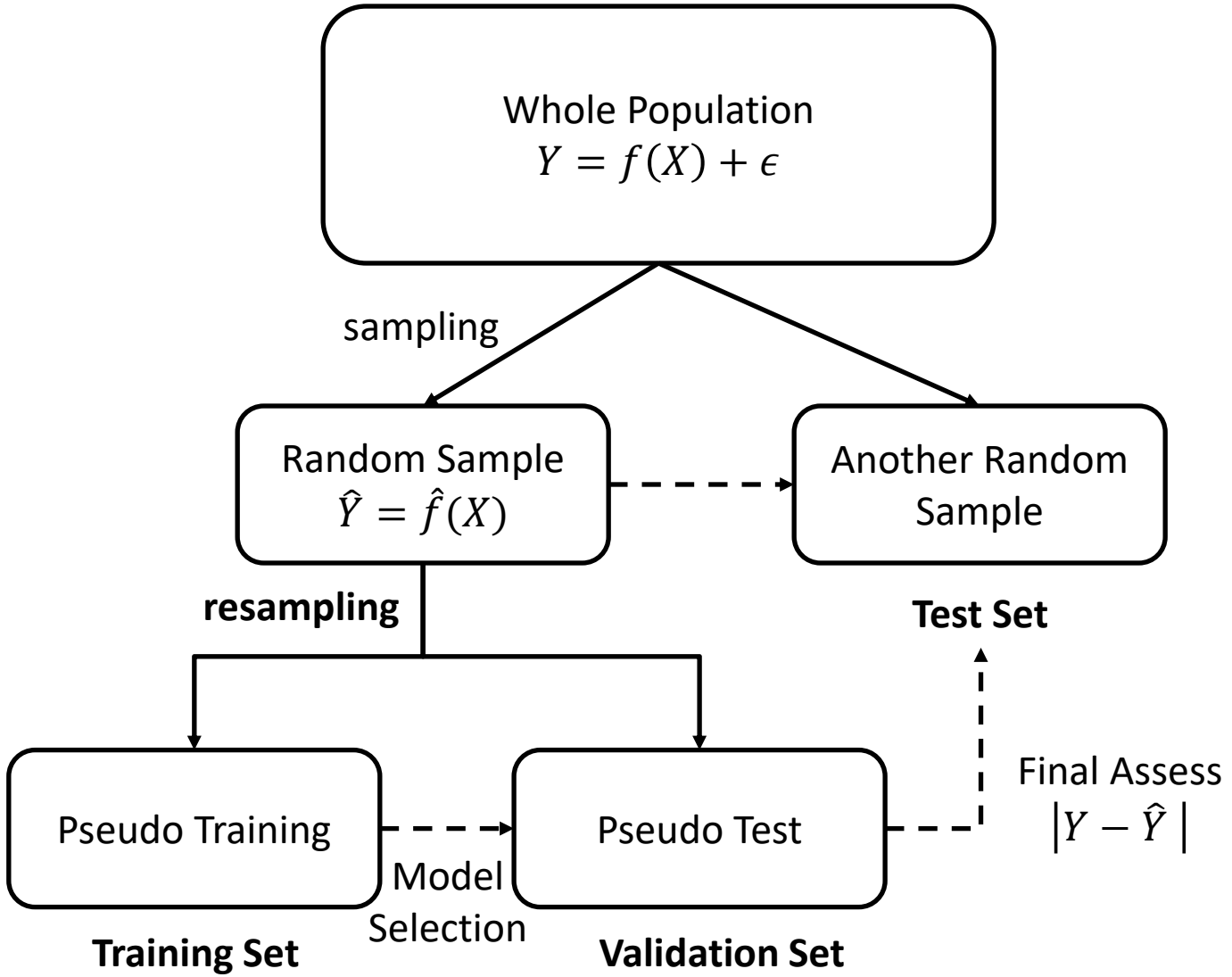
학습 문제의 설정

- 학습의 목표: 임의로 추출된 훈련 데이터에서 모집단의 $f(X)$ 를 추정
  - 평가 데이터를 보지 않고 훈련 데이터에서 성능을 최대화하는 모델을 찾는 것



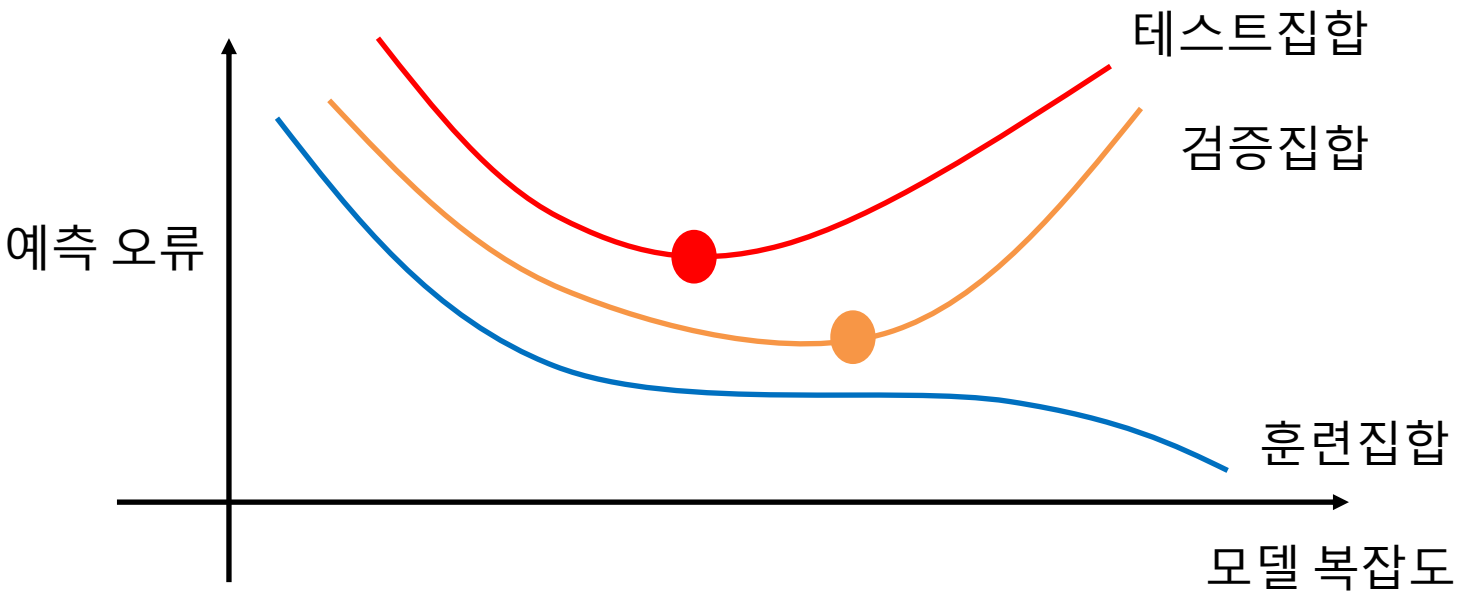
검증 집합 (Validation Set)

- 훈련-평가 데이터 집합의 관계를 훈련 데이터 내에서 재표집을 통해 시뮬레이션



모델의 복잡도에 따른 성능의 변화

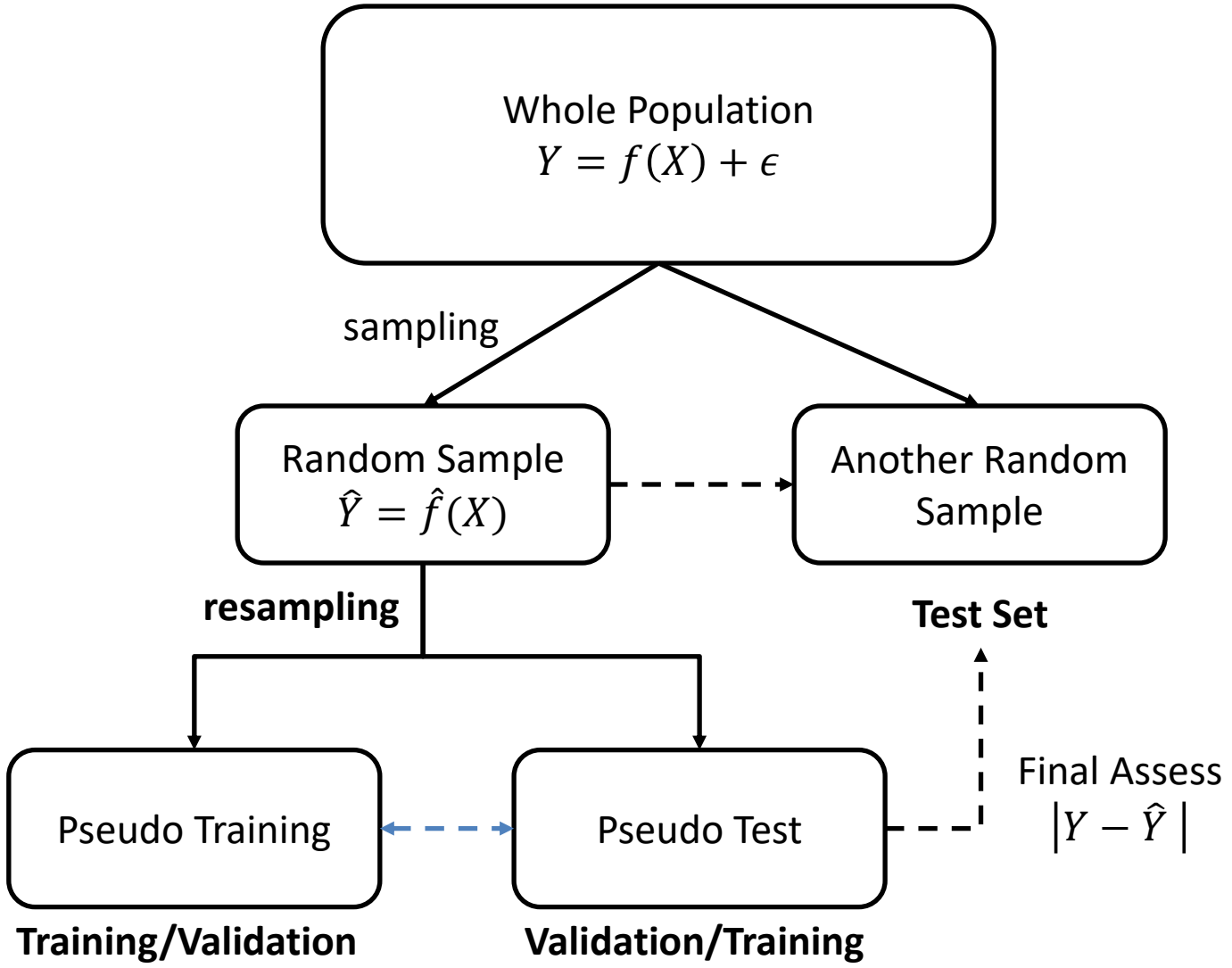
- 모델의 복잡도(Complexity)에 따른 성능의 변화



- 검증 에러는 실제 테스트 에러의 패턴을 잘 보여줌
  - 검증 집합을 통해 최적의 모델을 선택하는 것이 가능
  - 실제로 검증 에러는 실제 테스트 에러보다 작은 경향을 보임
  - 일반적으로 검증집합에서의 성능은 최종적인 모델의 성능으로 취급될 수 없음
  - 모델의 최종 성능은 반드시 테스트 집합에서의 성능을 통해 측정

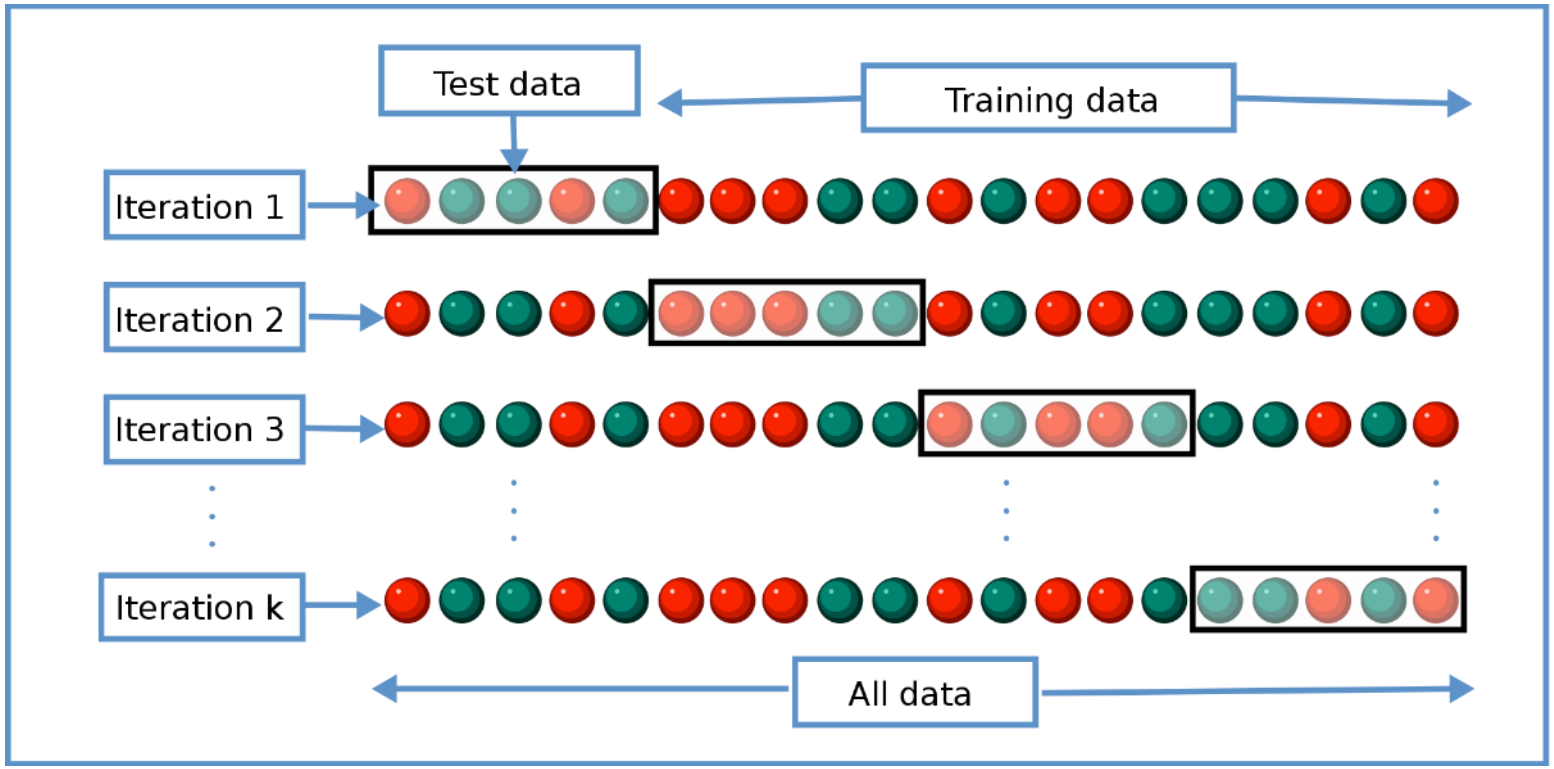
교차 검증 (CV: Cross Validation)

- 검증집합을 이용하는 방식과 유사하지만, 훈련집합과 검증집합이 서로 바뀔 수 있음



K-폴드 교차 검증 (K-Fold Cross Validation)

- 전체 훈련집합을 K개의 그룹으로 나눈 후, 하나의 그룹을 검증집합 나머지 그룹을 훈련집합으로 생각하여 검증을 진행. 이것을 각각의 그룹에 대하여 K번 반복
- K개의 검증집합의 성능을 평균하여 교차검증 성능을 측정



- LOOCV (Leave-One-Out) 교차검증: n-폴드 교차검증
  - 하나의 그룹이 하나의 샘플만을 포함

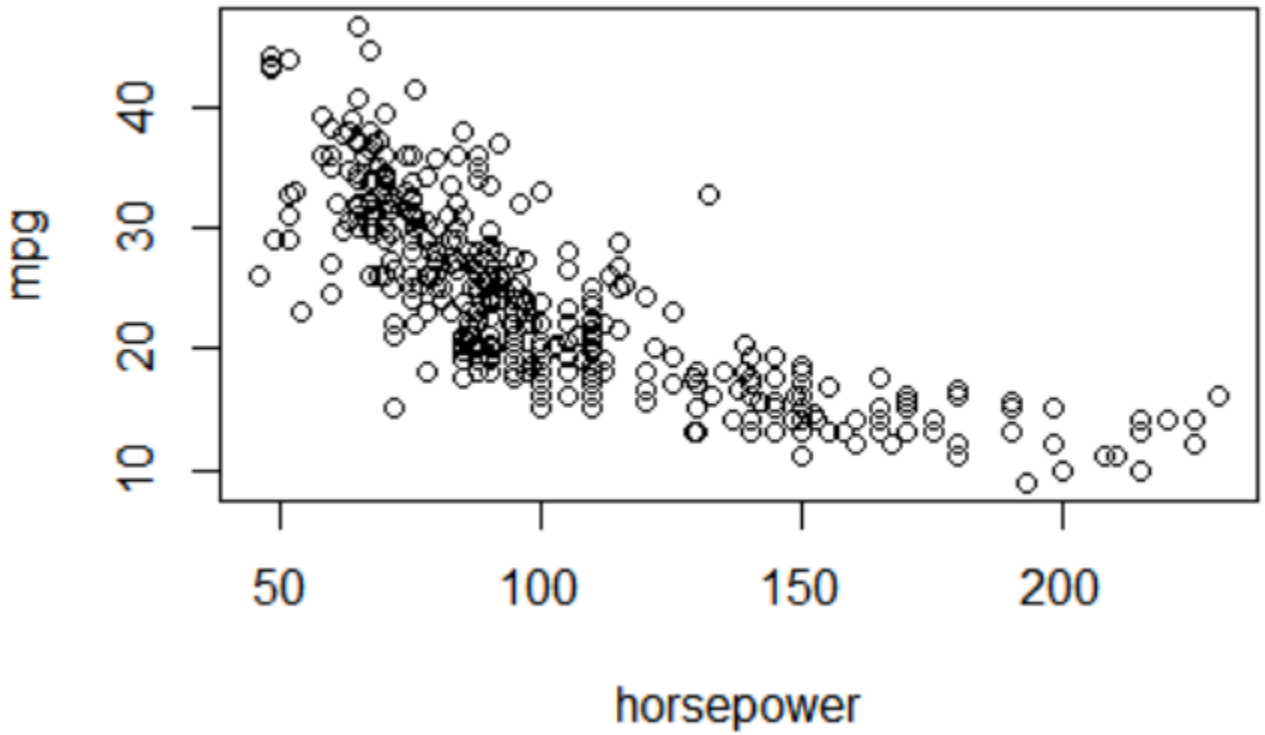
검증 집합 vs. 교차 검증

- 총 1,000 개의 데이터가 있을 때….
  - 500개의 훈련 데이터와 500개의 검증 데이터로 분리
    - 500개의 데이터로 단 1번 훈련
  - LOOCV
    - 999개의 데이터로 1,000 번 훈련

- 총 20 개의 데이터가 있을 때….
  - 10개의 훈련 데이터와 10개의 검증 데이터로 분리
    - 10개의 데이터로 단 1번 훈련
  - LOOCV
    - 19개의 데이터로 20 번 훈련

예제

- 자동차 데이터 셋
  - Output: mpg, input: horsepower
  - 261 training and 131 test samples



예제

- 다항 회귀 모델
  - 1차~4차 모델 중 하나를 선정
  - $Y \sim \beta_0 + \beta_1 X, \ldots, Y \sim \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4$

- 검증집합 접근법: 50% / 50%
- 교차검증법: LOOCV & 5-fold CV

- 결과

	1st	2nd	3rd	4th
Train MSE	23.10	19.51	19.50	**19.43**
Validation MSE	22.20	**16.28**	16.52	16.49
LOOCV MSE	23.49	**19.89**	20.05	20.19
5-fold CV MSE	23.57	**19.68**	19.90	20.19
Test MSE		**18.44**		

모델 확장

모델 확장

- 기본적인 모델들은 다양한 방식으로 확장이 가능
  - 우리가 사용할 수 있는 모델을 풍부하게 만듦

- 입력 변수 선택 (Feature Selection)
  - 활용가능한 입력 변수 중 일부를 골라서 사용
  - 모든 모델에 적용 가능
  - 복잡성을 줄이는 역할
- 차원 축소 (Dimension Reduction)
  - 입력 변수의 차원을 더 낮은 차원으로 압축하여 사용
  - 연속형 입력 변수에 대하여 사용 가능
  - 복잡성을 줄이는 역할
- 규제화 (Regularization)
  - 모델 파라메터의 범위를 제한
  - 모델 파라메터가 존재하는 모수적 모델에 적용 가능
  - 복잡성을 줄이는 역할
- 커널 확장 (Kernel Expansion)
  - 입력 변수의 차원을 더 높은 차원으로 확장하여 사용
  - 복잡성을 높이는 역할

입력 변수 선택 (Feature Selection)

- 어떤 모델에 대하여 p개의 주어진 입력 변수 중 가장 좋은 k개의 변수를 선택
  - 몇 개가 좋을지 어떤 변수들이 좋을지 모두 선택해야 함

- 최적의 해법 (Best selection)
  - 모든 경우에 대하여 조사해보고 검증집합에서 가장 좋은 성능을 갖는 모델을 선택
  - p에 따라 경우가 기하급수적으로 증가하여 실질적으로 적용 불가능

- 예제: 3개의 입력변수가 주어진 선형회귀모델
  - $k = 0$: $Y \approx \beta_0$
  - $k = 1$: $Y \approx \beta_0 + \beta_1 X_1$, $Y \approx \beta_0 + \beta_1 X_2$, $Y \approx \beta_0 + \beta_1 X_3$
  - $k = 2$: $Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2$, $Y \approx \beta_0 + \beta_1 X_2 + \beta_2 X_3$, $Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_3$
  - $k = 3$: $Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$

입력 변수 선택 (Feature Selection)

- Stepwise Selection
  - 한 번에 모든 조합을 찾는 것이 아니라 그 전 단계에서 가장 좋은 조합으로부터 탐색을 시작

- Forward Stepwise Selection
  - $k = 1$:가장 좋은 변수를 하나 선택
  - $k = 2$: $k = 1$에서 선택된 가장 좋은 변수에 추가할 변수를 하나 선택
  - $k = 3$: $k = 2$에서 선택된 2개의 변수에 추가할 변수를 선택
  - 각 단계에서 찾아진 모델 중 전체적으로 가장 좋은 모델을 선택

	X1	X2	X3	X4	X5
k=0					
k=1					
k=2					
k=3					
k=4					
k=5					

입력 변수 선택 (Feature Selection)

- **Backward Stepwise Selection**
  - $k = p$: 모든 변수를 이용하여 모델을 학습
  - $k = p - 1$: $k = p$에서 하나를 제거했을 때 성능이 가장 좋은 변수를 선택
  - $k = p - 2$: $k = p - 1$에서 가장 불필요한 변수를 선택해서 제거
  - 각 단계에서 찾아진 모델 중 전체적으로 가장 좋은 모델을 선택

	X1	X2	X3	X4	X5
k=5					
k=4					
k=3					
k=2					
k=1					
k=0					

- Forward와 Backward는 반드시 같은 결과를 내지 않음.
- Backward가 계산량이 많기 때문에 보통 Forward만 쓰거나 F/B을 같이 씀

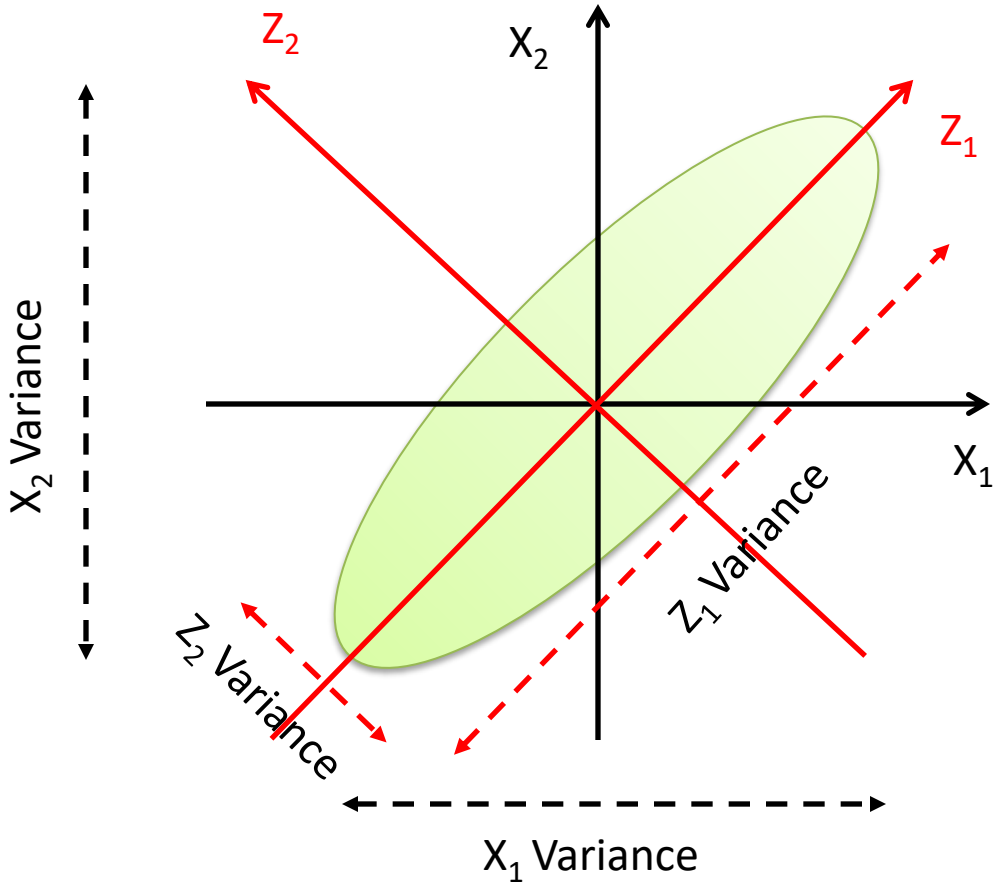
차원 축소 (Dimension Reduction)

- 차원축소: 주어진 p차원의 입력 변수 공간 $(X_1, X_2, \dots, X_p)$를 더 낮은 k 차원의 공간$(Z_1, Z_2, \dots, Z_k)$으로 변환
  - 차원의 저주를 피하기 위하여 (고차원 모델에서는 어떤 모델도 적합하지 않음)
  - 불필요한 정보를 제거 (예: X_1과 X_2 가 비슷한 경우)
  - 새로운 입력 변수(feature)를 추출
  - 시각화 (2/3차원 공간만 시각화 가능)

- 변수 선택 (Feature Selection): 주어진 변수 중 일부를 선택
- 변수 추출 (Feature Extraction): 원래 변수에서 새로운 변수를 도출
  - 주성분 분석 (PCA: Principal Component Analysis)
  - ICA (Independent Component Analysis)
  - NMF (Non-negative Matrix Factorization)
  - 다양한 딥러닝 기법 (Autoencoder, Convolution Layer)

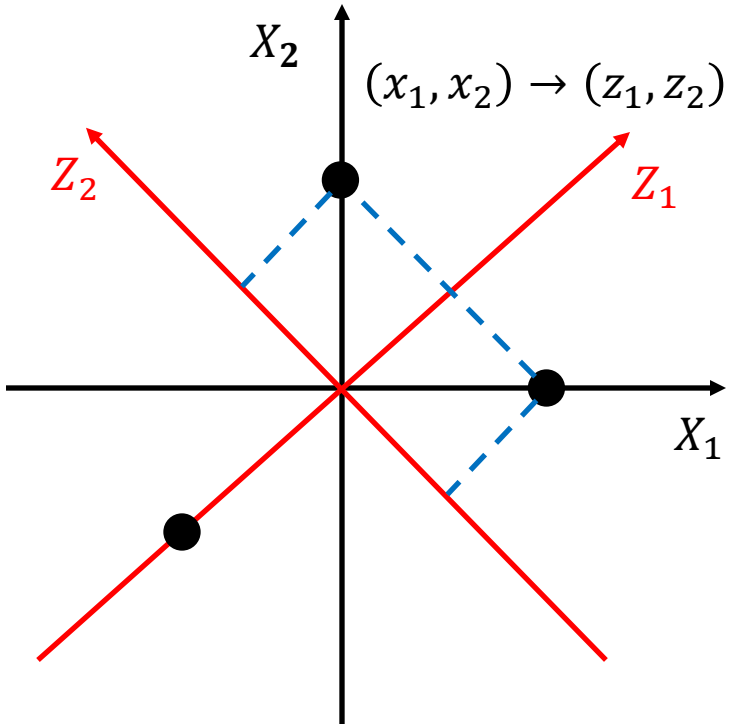
주성분 분석을 이용한 차원 축소

- 주성분 분석 (PCA: Principal Component Analysis)
  - 되도록 적은 수의 변수로 되도록 많은 데이터의 변화량을 설명하고 싶음
  - 데이터의 변화량을 가장 잘 설명하는 변수(주성분)를 차례로 추출
  - 나중에 추출된 변수를 제거하여 효과적으로 차원을 축소



주성분 분석을 이용한 차원 축소

- 주성분 분석 (차원축소에서 자세히 다룸)
  - 첫 번째 주성분 (PC1): 가장 많은 변화량을 설명하는 변수
  - 두 번째 주성분 (PC2): 그 다음 많은 변화량을 설명하는 변수
  - 주성분의 수는 변수의 개수만큼 찾을 수 있음
  - 좌표축 변환(=선형적 변환)으로 생각할 수 있음



	X_1	X_2
1	0	1
2	1	0
3	-1	-1

⇨

	Z_1	Z_2
1	$1/\sqrt{2}$	$1/\sqrt{2}$
2	$1/\sqrt{2}$	$-1/\sqrt{2}$
3	$-\sqrt{2}$	0

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} p_1^T \\ p_2^T \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

예제

원본 데이터

	X1	X2	X3	Y
0	1.39	0.84	0.64	1.82
1	0.60	1.76	0.07	2.26
2	0.68	1.41	0.25	-0.58
3	-2.84	-1.94	-0.79	-3.96
4	-1.88	-1.25	-0.96	-0.56
5	-0.28	-2.63	-1.51	-0.99
6	-0.34	2.06	0.05	1.27
7	-0.73	0.35	-2.08	-2.12

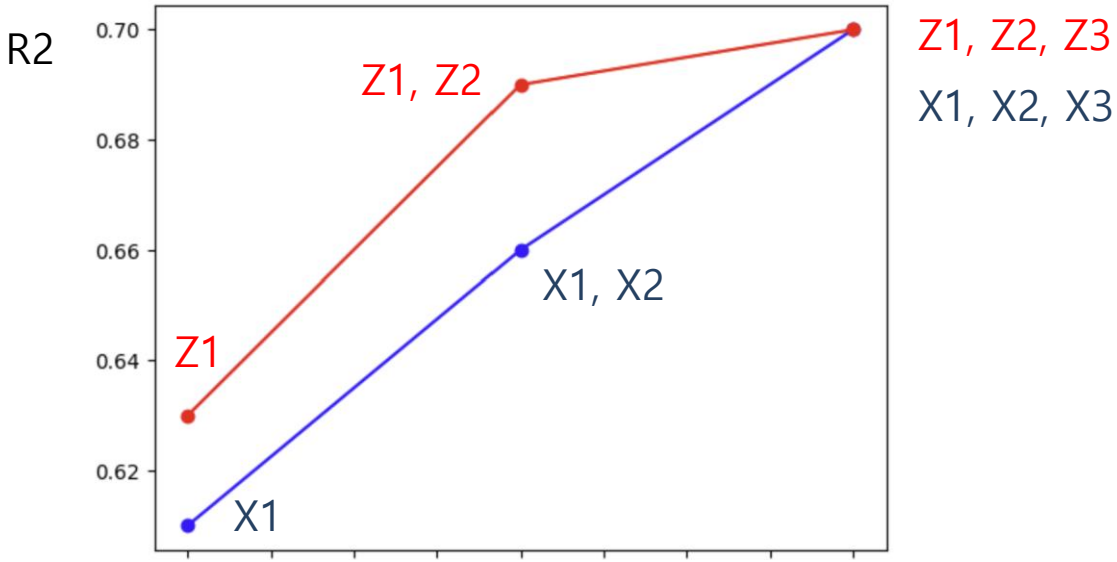
주성분 변환 데이터

	Z1	Z2	Z3	Y
0	1.95	1.09	0.53	1.82
1	2.06	-0.11	-0.05	2.26
2	1.89	0.17	0.19	-0.58
3	-2.95	-0.82	0.78	-3.96
4	-1.95	-0.44	0.24	-0.56
5	-2.35	1.63	-0.30	-0.99
6	1.79	-1.05	0.07	1.27
7	-0.44	-0.48	-1.45	-2.12

Y ~ X1+X2+X3

vs.

Y ~ Z1+Z2+Z3



주성분 분석을 이용한 차원 축소

- 주성분 분석을 학습 모델에 적용
  - 원 데이터를 주성분 분석을 이용해 변환 (X -> Z)
  - 원 모델: $Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$
  - 첫 번째 주성분을 이용한 모델 $Y \approx \beta_0 + \beta_1 Z_1$
  - 두 번째 주성분까지 이용한 모델 $Y \approx \beta_0 + \beta_1 Z_1 + \beta_2 Z_2$
  - 세 번째 주성분까지 이용한 모델 $Y \approx \beta_0 + \beta_1 Z_1 + \beta_2 Z_2 + \beta_3 Z_3$
- 몇 가지 이슈
  - 모든 주성분을 이용한 모델은 원 모델과 같은가?
  - 주성분을 차례로 이용하지 않아도 괜찮은가?
  - 주성분 분석이 언제 유리한가?

규제화 (Regularization)

- 모델 파라메터의 범위를 제한함으로써 모델을 복잡성을 줄이는 것이 가능

(1) $Y \approx \beta_0 + \beta_1 X$ for $-\infty < \beta_1 < \infty$
(2) $Y \approx \beta_0 + \beta_1 X$ for $-100 < \beta_1 < 100$
(3) $Y \approx \beta_0 + \beta_1 X$ for $-1 < \beta_1 < 1$
(4) $Y \approx \beta_0 + \beta_1 X$ for $-10^{-10} < \beta_1 < 10^{-10}$

- 규제화의 구현 (선형 회귀의 경우)
  - 일반적 선형 회귀의 손실 함수

$$L = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

  - 규제화가 적용된 손실 함수

벌점 (Penalty Term)

$$L = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 + \lambda \beta_1^2$$

규제화 (Regularization)

- 기계학습에서 규제화가 적용된 손실 함수는 다음과 같이 표현 가능

$$\text{Loss}(\theta, \lambda | X, Y) = \text{Error}(\theta | X, Y) + \lambda \text{Penalty}(\theta)$$

  - 에러 함수는 제곱에러, 크로스 엔트로피, 힌지 로스 등이 가능
  - 벌점은 L_0, L_1, L_2, $L_{infinite}$ 등이 가능
  - θ 는 모델 파라메터로 훈련 과정에서 최적화 됨
  - λ 는 튜닝 파라메터로 우리가 최적의 성능을 얻기 위해 결정

- 규제화는 과최적화를 방지하는 핵심적인 기법

- 예제: 로지스틱 회귀에 L2 규제화 추가

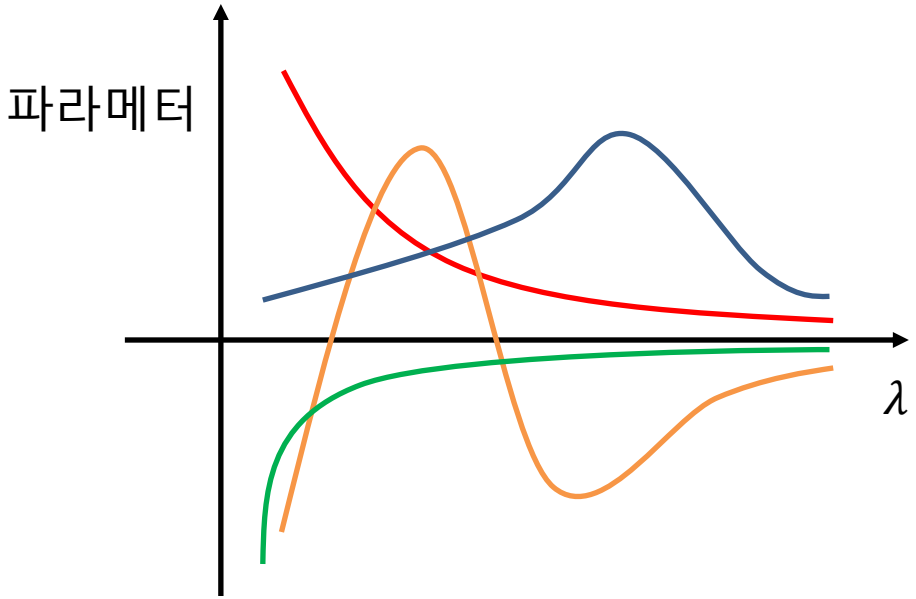
$$Loss = -\sum_i \left[y_i \log\left(\frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}} \right) + (1 - y_i) \log\left(\frac{1}{1 + e^{\beta_0 + \beta_1 x}} \right) \right] + \lambda \beta_1^2$$

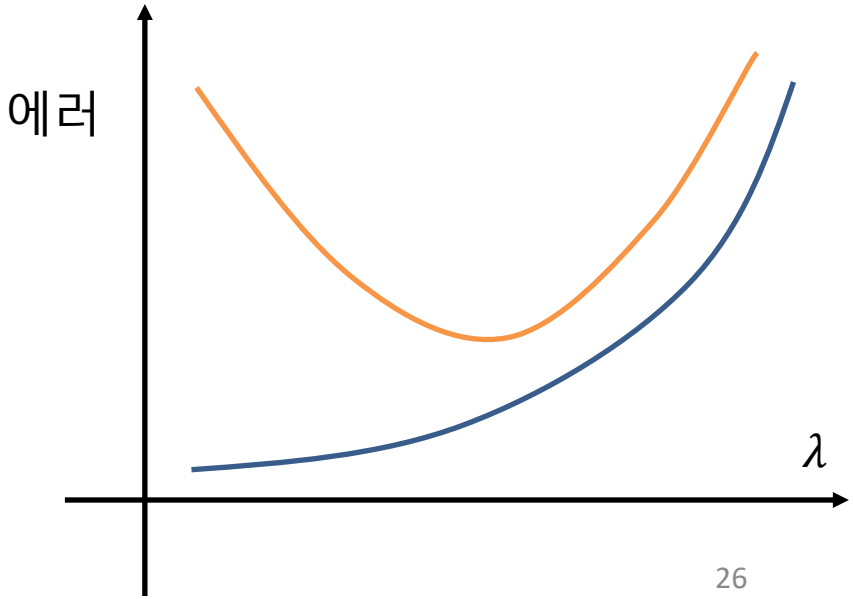
규제화 (Regularization)

- L_2 규제화 (Tikhonov regularization)
  - $\text{Penalty}(\theta) = \theta_1^2 + \theta_2^2 + \cdots + \theta_p^2$
  - 규제가 강해질 수록 각 파라메터가 0으로 접근 (하지만 0이 되지는 않음)

- Ridge Regression: 선형회귀 + L_2 규제화

$$L = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{i=1}^{p} \beta_i x_i \right)^2 + \lambda \sum_{i=1}^{p} \beta_i^2$$





규제화 (Regularization): Ridge Regression 예제

- 데이터

	X_1	X_2	Y
1	0	0	1
2	1	0	2
3	0	1	3
4	1	1	3

모델: $Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2$

일반 선형 회귀 모델

$\hat{y} = 1.25 + 0.5x_1 + 1.5x_2$

- 손실함수

$$L = \sum_{i=1}^{4}(y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2})^2 + \lambda(\beta_1^2 + \beta_2^2)$$
$$= (1 - \beta_0)^2 + (2 - \beta_0 - \beta_1)^2 + (3 - \beta_0 - \beta_2)^2 + (3 - \beta_0 - \beta_1 - \beta_2)^2 + \lambda(\beta_1^2 + \beta_2^2)$$

- 손실함수 최소화

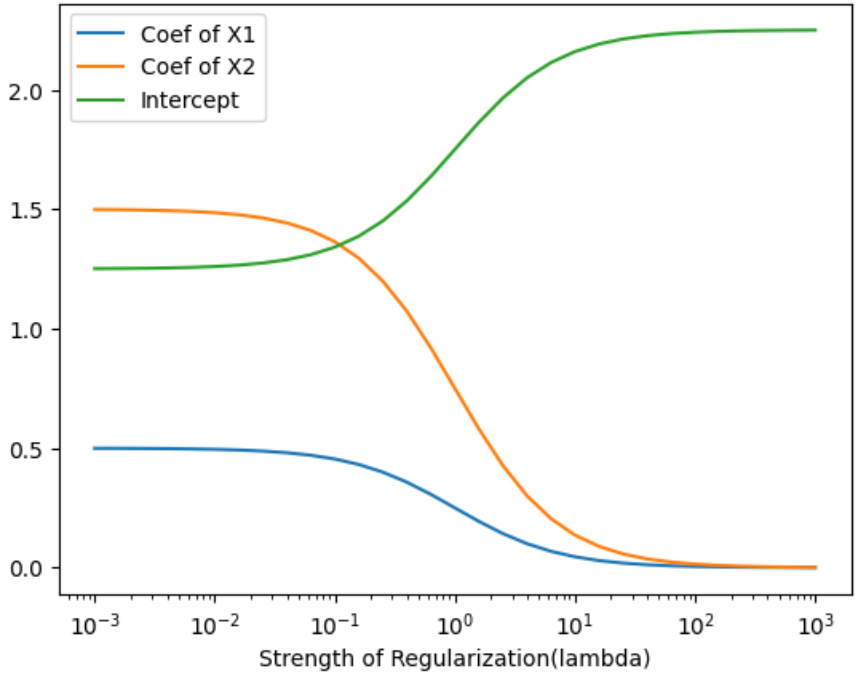
$$\frac{\partial L}{\partial \beta_0} = -2(1 - \beta_0) - 2(2 - \beta_0 - \beta_1) - 2(3 - \beta_0 - \beta_2) - 2(3 - \beta_0 - \beta_1 - \beta_2) = 0$$
$$\frac{\partial L}{\partial \beta_1} = -2(2 - \beta_0 - \beta_1) - 2(3 - \beta_0 - \beta_1 - \beta_2) + 2\lambda\beta_1 = 0$$
$$\frac{\partial L}{\partial \beta_2} = -2(3 - \beta_0 - \beta_2) - 2(3 - \beta_0 - \beta_1 - \beta_2) + 2\lambda\beta_2 = 0$$

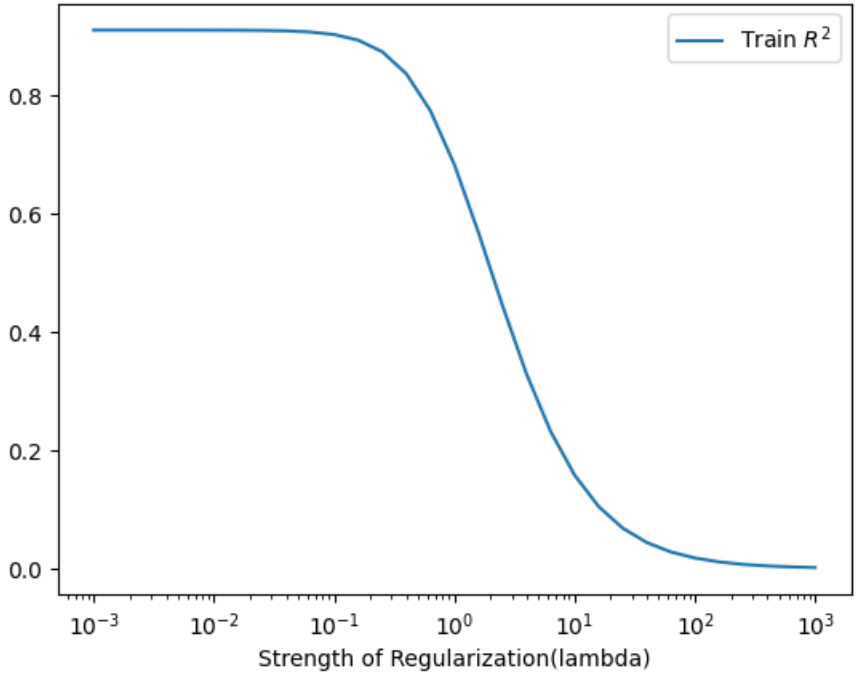
규제화 (Regularization): Ridge Regression 예제

- 해를 구하면…

$$\beta_0 = 1.25 + \left(1 - \frac{1}{1-\lambda}\right), \quad \beta_1 = \frac{0.5}{1-\lambda}, \quad \beta_2 = \frac{1.5}{1-\lambda}$$

- 규제 정도에 따른 계수와 Train R2의 변화



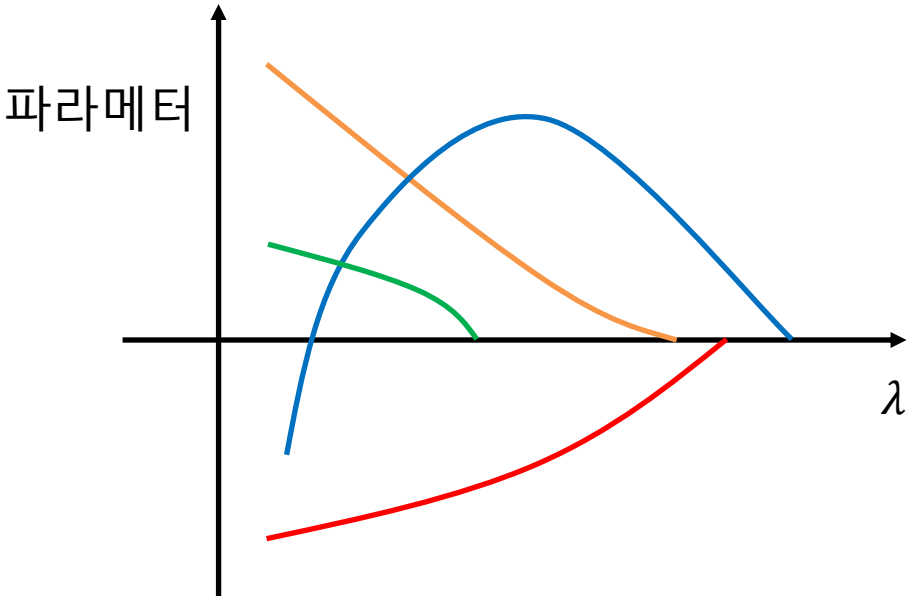


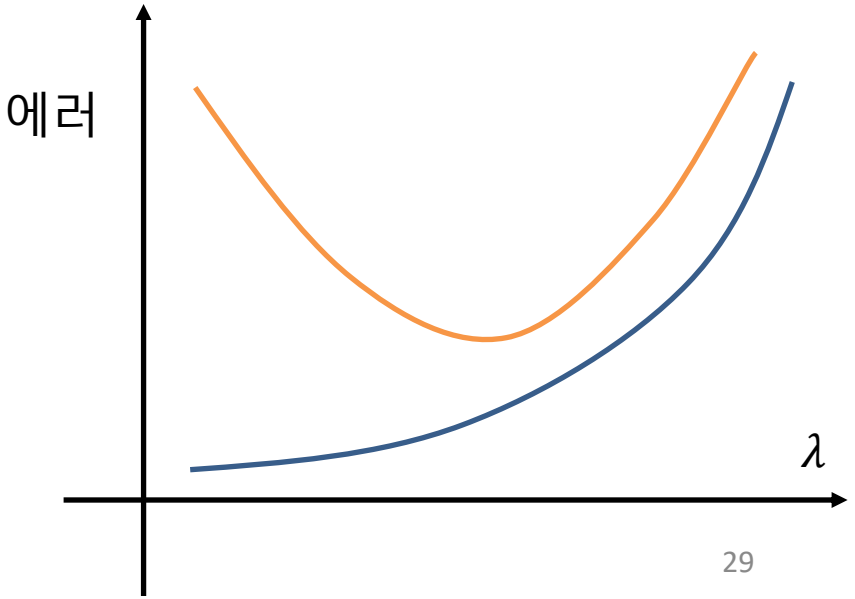
규제화 (Regularization)

- L_1 규제화
  - $\text{Penalty}(\theta) = |\theta_1| + |\theta_2| + \cdots + |\theta_p|$
  - 규제가 강해질 수록 각 파라메터가 0으로 접근하다가 0이 됨
  - 변수 선택이 가능!!!!

- Lasso Regression: 선형회귀 + L_1 규제화

$$L = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{i=1}^{p} \beta_i x_i \right)^2 + \lambda \sum_{i=1}^{p} |\beta_i|$$





규제화 (Regularization)

- Elastic Net

$$L = ERROR(\theta|X,Y) + \lambda_1 \sum_{i=1}^{p} |\theta_i| + \lambda_2 \sum_{i=1}^{p} \theta_i^2$$

$$L = ERROR(\theta|X,Y) + \alpha \left(\lambda \sum_{i=1}^{p} |\theta_i| + \frac{1}{2}(1-\lambda) \sum_{i=1}^{p} \theta_i^2 \right)$$

- L_0 규제화: 0이 아닌 파라메터의 수 (i.e. 변수 선택의 일종)
- $L_{infinite}$ 규제화: 파라메터 중 최대값

- 각종 규제화 기법은 딥러닝 모델의 과대적합을 방지하기 위해서도 주요하게 사용됨

KOREA UNIVERSITY

감사합니다