

# 모델 일반화, 편향-분산 트레이드오프, 그리고 정규화 (Ridge & Lasso)

CS109A: Introduction to Data Science (강사: Pavlos Protopapas)  
내용 통합 및 재구성 노트

October 26, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 07
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 07의 핵심 개념 학습

## Contents

1 개요 . . . . .	2
2 용어 정리 . . . . .	3
3 핵심 개념: 편향-분산 트레이드오프 (Bias-Variance Tradeoff) . . . . .	4
3.1 모델 오차의 두 가지 근원 . . . . .	4
3.2 편향(Bias)과 분산(Variance)의 정의 . . . . .	4
3.3 트레이드오프(Tradeoff) 관계 . . . . .	5
4 문제 진단: 과대적합과 모델 계수 . . . . .	6
5 해결책: 정규화 (Regularization) . . . . .	7
5.1 정규화의 핵심 아이디어 . . . . .	7
5.2 $\lambda$ (람다)의 역할: 패널티의 강도 . . . . .	7
5.3 두 가지 정규화 기법: L2 (Ridge) vs. L1 (Lasso) . . . . .	7
5.4 계산적 차이 및 비교 요약 . . . . .	8
6 절차: 최적의 $\lambda$ (람다) 찾기 (Hyperparameter Tuning) . . . . .	9
6.1 방법 1: 단일 검증 세트 (Single Validation Set) 사용 . . . . .	9
6.2 방법 2: K-Fold 교차 검증 (Cross-Validation) . . . . .	10
7 FAQ 및 주요 질문 . . . . .	11
8 1페이지 요약: 빠른 훑어보기 . . . . .	12

## 1 개요

이 문서는 머신러닝 모델의 성능을 평가하고 개선하는 핵심 원리인 \*\*일반화(Generalization)\*\*, \*\*편향-분산 트레이드오프(Bias-Variance Tradeoff)\*\*, 그리고 \*\*정규화(Regularization)\*\* 기법에 대해 다룹니다.

우리의 최종 목표는 훈련(training) 데이터에만 잘 맞는 모델이 아니라, 한 번도 본 적 없는 새로운 데이터(test data)에서도 좋은 성능을 내는 \*\*'일반화' 성능이 뛰어난\*\* 모델을 만드는 것입니다.

모델이 너무 단순하면 훈련 데이터조차 제대로 학습하지 못하며(과소적합, High Bias), 모델이 너무 복잡하면 훈련 데이터의 노이즈까지 암기해버려 새로운 데이터에서 형편없는 성능을 보입니다(과대적합, High Variance).

이 문서는 과대적합의 주된 증상(모델 계수의 폭주)을 진단하고, 이를 해결하기 위해 손실 함수(Loss Function)에 \*\*'패널티'\*\*를 부과하는 정규화 기법, 특히 \*\*릿지(Ridge, L2)\*\*와 \*\*라쏘(Lasso, L1)\*\*를 중점적으로 설명합니다.

마지막으로, 이 패널티의 강도를 조절하는 하이퍼파라미터 \*\* $\lambda$  (람다)\*\*를 찾기 위한 체계적인 절차로 \*\*검증 세트(Validation Set)\*\*와 \*\*교차 검증(Cross-Validation)\*\* 방법을 단계별로 학습합니다.

## 2 용어 정리

핵심 용어들을 미리 이해하면 학습에 도움이 됩니다.

**Table 1:** 핵심 용어 정리표

용어	원어 (Full Term)	쉬운 설명 (초심자용)	비고
일반화	Generalization	모델이 훈련 데이터가 아닌 '새로운 데이터'를 얼마나 잘 맞추는지의 능력.	높을수록 좋은 모델입니다.
과소적합	Underfitting	모델이 너무 단순해서 훈련 데이터조차 제대로 학습하지 못한 상태.	편향(Bias)이 높은 상태입니다.
과대적합	Overfitting	모델이 너무 복잡해서 훈련 데이터의 '노이즈'까지 암기해버린 상태.	새로운 데이터에서 성능이 급격히 저하됩니다. 분산(Variance)이 높은 상태입니다.
편향	Bias	모델의 예측이 '실제 정답'과 평균적으로 얼마나 멀리 떨어져 있는가.	'부정확성'. 과녁의 중심을 못 맞춤.
분산	Variance	훈련 데이터가 조금 바뀔 때 모델의 예측이 얼마나 크게 출렁이는가.	'비일관성'. 쏠 때마다 탄착군이 흘어짐.
정규화	Regularization	모델의 복잡도(주로 계수 값)에 패널티를 부과하여 과대적합을 막는 기법.	"모델이 너무 복잡해지지 마!"
릿지 (L2)	Ridge Regression	계수의 '제곱의 합'에 패널티를 주는 정규화. ( $L_2$ Norm)	계수를 0에 가깝게 줄이지만 0으로 만들진 않음.
라쏘 (L1)	Lasso Regression	계수의 '절대값의 합'에 패널티를 주는 정규화. ( $L_1$ Norm)	불필요한 계수를 아예 0으로 만들어 '변수 선택' 효과.
$\lambda$ (람다)	Lambda	정규화의 '강도'를 조절하는 하이퍼파라미터.	0이면 정규화 안함. $\infty$ 면 모든 계수가 0이 됨.
하이퍼파라미터	Hyperparameter	모델이 스스로 학습하는 값( $\beta$ )이 아니라, '사람이 직접 설정'해줘야 하는 값. $\lambda$ 나 K-Fold의 $K$ 값 등.	

### 3 핵심 개념: 편향-분산 트레이드오프 (Bias-Variance Tradeoff)

모델의 예측 오차(Error)는 우리가 어떻게 할 수 없는 부분과, 우리가 개선할 수 있는 부분으로 나뉩니다.

#### 3.1 모델 오차의 두 가지 근원

##### 1. 환원 불가능한 오차 (Irreducible Error)

이 오차는 데이터 자체에 내재된 \*\*'무작위 노이즈'\*\* 때문에 발생합니다. 아무리 완벽한 모델을 만들어도 이 오차는 절대 0이 될 수 없습니다.

###### □ 예제:

\*\*비유 (Aleatoric Error):\*\* 아무리 비싼 최고급 마이크( $\approx$ 모델)를 사용해도, 녹음실 주변의 공사장 소음( $\approx$ 노이즈)까지 함께 녹음되는 것과 같습니다. 이 소음은 마이크 성능으로 제거할 수 없습니다.

우리는 이 오차의 존재를 인정하고, 우리가 줄일 수 있는 오차에 집중해야 합니다.

##### 2. 환원 가능한 오차 (Reducible Error)

이 오차는 우리가 \*\*모델을 잘못 선택\*\*했기 때문에 발생합니다. 우리의 임무는 이 오차를 최소화하는 것이며, 이 오차는 다시 '편향'과 '분산'이라는 두 가지 요소로 분해됩니다.

#### 3.2 편향(Bias)과 분산(Varianc)e의 정의

모델의 성능을 사격에 비유하여 편향과 분산을 이해할 수 있습니다.

##### 편향 (Bias): 과녁을 놓치다 (부정확성)

\*\*편향\*\*은 모델의 예측 값이 실제 정답(과녁의 중심)과 평균적으로 얼마나 멀리 떨어져 있는지를 나타냅니다.

- **High Bias (높은 편향):** 모델이 너무 단순하여 데이터의 복잡한 패턴을 전혀 학습하지 못합니다. (예: S자 곡선 데이터를 직선으로 예측하려는 시도)
- **결과: 과소적합 (Underfitting).** 훈련 데이터에서도, 테스트 데이터에서도 모두 성능이 나쁩니다.
- **특징:** 훈련 데이터를 바꿔가며 여러 번 학습해도 예측 결과가 거의 변하지 않습니다 (낮은 분산).

### 분산 (Variance): 탄착군이 흩어지다 (비일관성)

\*\*분산\*\*은 훈련 데이터가 조금만 바뀌어도 모델의 예측이 얼마나 크게 변동하는지를 나타냅니다.

- **High Variance (높은 분산):** 모델이 너무 복잡하여 훈련 데이터의 사소한 노이즈까지 '암기'해버립니다.
- **결과: 과대적합 (Overfitting).** 훈련 데이터에서는 완벽(0에 가까운 오차)하지만, 새로운 테스트 데이터에서는 성능이 재앙 수준입니다.
- **특징:** 훈련 데이터 샘플이 조금만 달라져도 모델의 형태가 스파게티 면발처럼 마구 요동칩니다.

#### □ 예제:

\*\*시뮬레이션 예시 (스파게티 면발):\*\* 서로 다른 2,000개의 샘플 데이터셋을 뽑아서 모델을 2,000번 학습시켰다고 가정해봅시다.

- **단순한 선형 모델 (Low Variance):** 2,000개의 예측선이 모두 비슷하게 그려집니다. (안정적)
- **복잡한 10차 다항식 모델 (High Variance):** 2,000개의 예측선이 샘플 데이터의 노이즈에 민감하게 반응하여, 마치 스파게티 면발처럼 서로 얹히고설켜 그려집니다. (불안정)

### 3.3 트레이드오프(Tradeoff) 관계

편향과 분산은 한쪽이 줄어들면 다른 한쪽이 늘어나는 \*\*'시소 관계'\*\*에 있습니다.

#### 모델 복잡도에 따른 오차의 변화

##### 모델 복잡도(X축) vs. 총 오차(Y축)

- **모델이 단순할수록 (좌측):**
    - 편향(Bias)은 높고 (데이터를 못 맞춤)
    - 분산(Variance)은 낮습니다. (예측이 안정적)
  - **모델이 복잡해질수록 (우측):**
    - 편향(Bias)은 낮아지고 (훈련 데이터를 완벽히 맞춤)
    - 분산(Variance)은 급격히 높아집니다. (데이터에 과민 반응)
- 총 오차(Total Error = Bias<sup>2</sup> + Variance + Irreducible Error)는 U자 형태의 곡선을 그립니다. 우리의 목표는 이 U자 곡선의 가장 낮은 지점, 즉 총 오차를 최소화하는 최적의 복잡도를 찾는 것입니다.

#### 사격 과녁 비유 요약표

**Table 2:** 편향과 분산의 4가지 시나리오 (사격 비유)

	Low Variance (낮은 분산 / 일관성 ↑)	High Variance (높은 분산 / 일관성 ↓)
High Bias (높은 편향 / 정확도 ↓)	<b>과소적합 (Underfitting)</b> <ul style="list-style-type: none"> <li>• 과녁의 중심은 못 맞추지만, 쏜 지점에 계속 일관되게 쏘.</li> <li>• (예: 단순 선형 모델)</li> </ul>	<b>최악의 모델 (Worst)</b> <ul style="list-style-type: none"> <li>• 과녁의 중심도 못 맞추고, 쏠 때마다 아무데나 흘어짐.</li> </ul>
Low Bias (낮은 편향 / 정확도 ↑)	<b>이상적인 모델 (Ideal)</b> <ul style="list-style-type: none"> <li>• 과녁의 중심(정답)에 정확하고 일관되게 쏘.</li> <li>• 우리의 목표!</li> </ul>	<b>과대적합 (Overfitting)</b> <ul style="list-style-type: none"> <li>• 평균적으로 과녁 중심 근처에 맞지만(훈련 데이터), 쏠 때마다 탄착군이 너무 흘어짐.</li> <li>• (예: 고차 다항식 모델)</li> </ul>

## 4 문제 진단: 과대적합과 모델 계수

질문: ”모델이 과대적합(High Variance)되었는지 어떻게 알 수 있습니까?”

답변: ”모델의 계수(coefficients,  $\beta$ ) 값을 확인하면 됩니다.”

모델이 과대적합 상태일 때, 즉 훈련 데이터의 노이즈에 과민하게 반응할 때, 모델의 \*\*계수( $\beta_j$ ) 값들은 비정상적으로 커지거나 극단적으로 불안정한 값\*\*을 갖게 됩니다.

### □ 예제:

\*\*계수 값 분포 비교 (Violin Plots):\*\* 서로 다른 2,000개의 샘플 데이터로 2,000개의 모델을 학습시킨 경우의 계수 분포입니다.

- **단순 선형 모델 (Low Variance):**  $\beta_0, \beta_1$  계수 값들이 0.0에서 1.25 사이의 좁은 범위에서 안정적으로 분포합니다.
- **복잡한 10차 다항식 모델 (High Variance):**  $\beta_5, \beta_8, \beta_9$  등의 고차항 계수 값들이  $1e9$  (즉, 10 억) 스케일로 폭주하며, 매우 넓은 범위(큰 분산)를 갖습니다.

### 주의사항

**과대적합의 핵심 증상:** 높은 분산(High Variance) → 불안정하고 극단적인 계수(Large  $\beta$ ) 값 따라서, 과대적합을 막기 위한 해결책은 ”모델의 계수 값이 너무 커지지 않도록 억제하는 것”입니다. 이것이 바로 ’정규화(Regularization)’의 핵심 아이디어입니다.

## 5 해결책: 정규화 (Regularization)

### 5.1 정규화의 핵심 아이디어

정규화는 모델의 손실 함수(Loss Function)를 수정하여, 모델이 두 가지 목표를 동시에 달성하도록 강제합니다.

1. **목표 1:** 데이터를 잘 맞춰라. (기존의 목표) → 손실 함수(MSE)를 최소화.  $\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
2. **목표 2:** 계수 값을 작게 유지해라. (새로운 목표: 과대적합 방지) → 계수의 크기에 대한 '패널티 항'을 최소화.

새로운 정규화 손실 함수:

$$\mathcal{L}_{\text{REG}} = (\text{기존 MSE}) + \lambda \times (\text{패널티 항})$$

$$\mathcal{L}_{\text{REG}} = \frac{1}{n} \sum_{i=1}^n (y_i - \beta^\top x_i)^2 + \lambda L_{\text{reg}}$$

모델은 이제 MSE만 줄이는 것이 아니라, (MSE +  $\lambda \times$  패널티)의 총합을 최소화해야 합니다.

### 5.2 $\lambda$ (람다)의 역할: 패널티의 강도

$\lambda$  (람다)는 패널티의 '강도'를 조절하는 하이퍼파라미터입니다.

- 만약  $\lambda = 0$  이라면: 패널티가 0이 됩니다. 이는 일반적인 선형 회귀와 같으며 과대적합의 위험이 있습니다.
- 만약  $\lambda \rightarrow \infty$  (무한대)라면: 패널티가 너무 강력해집니다. 모델은 MSE를 무시하고 오직 패널티 ( $\sum \beta_j^2$  또는  $\sum |\beta_j|$ )를 0으로 만드는 데만 집중합니다. 그 결과 모든 계수  $\beta_j$  가 0이 되어, 모델은 단순한 수평선(평균값)이 됩니다 (과소적합, High Bias).

#### ▣ 핵심 요약

우리의 목표는 훈련 데이터와 검증 데이터를 사용하여 편향과 분산 사이의 균형을 잡는 '최적의  $\lambda$ '를 찾는 것입니다.

### 5.3 두 가지 정규화 기법: L2 (Ridge) vs. L1 (Lasso)

패널티 항( $L_{\text{reg}}$ )을 어떻게 정의하느냐에 따라 릿지(Ridge)와 라쏘(Lasso)로 나뉩니다.

## 1. L2 정규화: 릿지 회귀 (Ridge Regression)

**패널티 항:** 계수의 \*\*제곱의 합\*\* ( $L_2$  Norm)  $\rightarrow L_{\text{reg}} = \sum_{j=1}^J \beta_j^2$   
**최종 손실 함수:**

$$\mathcal{L}_{\text{RIDGE}} = \frac{1}{n} \sum_{i=1}^n (y_i - \beta^\top x_i)^2 + \lambda \sum_{j=1}^J \beta_j^2$$

- 특징:** 계수 값이 커질수록 패널티가 '제곱'으로 증가하므로, 매우 큰(튀는) 계수 값을 강력하게 억제합니다. 모든 계수를 0에 '가깝게' 줄이지만, 정확히 0으로 만들지는 않습니다.
- 장점:** 계산이 빠릅니다 (수학적인 공식, 즉 'Analytical Solution'이 존재함). 다중공선성 (Multicollinearity: 예측 변수 간 강한 상관관계)이 있을 때 모델을 안정화시키는데 매우 효과적입니다.
- 적합한 상황:** 모든 변수(feature)가 예측에 어느 정도 기여한다고 판단될 때 사용합니다.

## 2. L1 정규화: 라쏘 회귀 (Lasso Regression)

**패널티 항:** 계수의 \*\*절대 값의 합\*\* ( $L_1$  Norm)  $\rightarrow L_{\text{reg}} = \sum_{j=1}^J |\beta_j|$   
**최종 손실 함수:**

$$\mathcal{L}_{\text{LASSO}} = \frac{1}{n} \sum_{i=1}^n (y_i - \beta^\top x_i)^2 + \lambda \sum_{j=1}^J |\beta_j|$$

- 특징:** 중요하지 않은 변수의 계수는 '정확히 0'으로 만들어 버립니다. 이는 모델에서 해당 변수를 아예 '제거'하는 것과 같은 효과를 줍니다.
- 장점:** 모델의 복잡도를 근본적으로 낮추는 \*\*자동 변수 선택(Feature Selection)\*\* 기능이 있습니다. 해석하기 쉬운 단순한 모델을 만듭니다.
- 적합한 상황:** 수백, 수천 개의 변수 중 실제 중요한 변수는 몇 개 안 된다고 의심될 때 매우 유용합니다.

### 주의사항

**주의:** 절편( $\beta_0$ )은 정규화하지 않습니다.  $L_1, L_2$  패널티 항은  $\beta_1$ 부터  $\beta_J$ 까지만 적용됩니다. 절편(intercept)  $\beta_0$ 는 특정 변수와 연결된 민감도가 아니라, 모델 전체의 기본 '높낮이(offset)'를 조절할 뿐이므로 패널티 대상에서 제외합니다.

## 5.4 계산적 차이 및 비교 요약

- 릿지(Ridge):** 행렬을 이용한 명확한 수학 공식(Analytical Solution)으로  $\beta$  값을 한 번에 계산할 수 있습니다.  

$$\hat{\beta}_{\text{Ridge}} = (X^\top X + \lambda I)^{-1} X^\top Y$$
- 라쏘(Lasso):** 절대값 함수는 0에서 미분이 불가능하므로, 이런 수학 공식이 없습니다. 대신 '수치적 최적화(Numerical Optimization)' 기법 (예: Solver, Coordinate Descent)을 사용하여 반복적으로  $\beta$  값을 찾아가야 하므로, 릿지보다 계산 속도가 느릴 수 있습니다.

**Table 3:** Ridge ( $L^2$ ) vs. Lasso ( $L^1$ ) 비교 요약

구분	릿지 회귀 (Ridge, $L^2$ )	라쏘 회귀 (Lasso, $L^1$ )
패널티 항	계수의 제곱의 합 ( $\sum \beta_j^2$ )	계수의 절대값의 합 ( $\sum  \beta_j $ )
계수 축소	계수를 0에 가깝게 줄임 (0은 안 됨)	불필요한 계수를 정확히 0으로 만듦
핵심 기능	모델 안정화, 다중공선성 제어	변수 선택 (Feature Selection)
계산 방식	빠름 (Analytical Solution 존재)	느림 (Numerical Solver 필요)
도형적 해석	패널티 영역이 '원' (Circle) 형태	패널티 영역이 '다이아몬드' (Diamond) 형태

## 6 절차: 최적의 $\lambda$ (람다) 찾기 (Hyperparameter Tuning)

$\lambda$ 는 하이퍼파라미터입니다. 즉, 모델이 스스로 학습하는 값이 아니라 우리가 정해줘야 하는 값입니다. 최적의  $\lambda$ 를 찾는 과정을 '튜닝(Tuning)'이라고 합니다.

### 주의사항

#### 하이퍼파라미터 튜닝의 철칙:

- 훈련(Train) 데이터로 튜닝하면 안 됩니다. (모델이  $\lambda = 0$ 을 선호하게 됨)
- 테스트(Test) 데이터로 튜닝하면 절대 안 됩니다. (정보 유출, 즉 'Cheating' 임)
- 오직 검증(Validation) 데이터 또는 교차 검증(Cross-Validation)을 사용해야 합니다.

### 6.1 방법 1: 단일 검증 세트 (Single Validation Set) 사용

가장 기본적인 방법으로, 데이터를 Train / Validation / Test 세 부분으로 나누어 진행합니다.

- Step 1:** 데이터 분할 (Split Data) 데이터를 훈련(Train), 검증(Validation), 테스트(Test)용으로 3-way 분할합니다.
- Step 2:**  $\lambda$  후보군 선정 (Select  $\lambda$  range) 테스트할  $\lambda$  값들의 목록을 만듭니다. (예: '[0.0001, 0.001, 0.01, 0.1, 1, 10, 100]')
- Step 3:** 모델 훈련 (Train Models) 훈련(Train) 데이터를 사용하여, 각  $\lambda$  후보마다 정규화 (Ridge/Lasso) 모델을 학습시키고 계수( $\beta_\lambda$ )를 얻습니다.
- Step 4:** 검증 및 MSE 기록 (Validate) 검증(Validation) 데이터를 사용하여, 3단계에서 얻은 각 모델( $\beta_\lambda$ )의 성능(MSE)을 측정합니다.

### 주의사항

**매우 중요!** 이 단계에서 성능을 측정할 때는 패널티 항( $\lambda L_{reg}$ )을 제외하고, 순수한 MSE 값만 계산합니다.

**이유:**  $\lambda$ 는 모델을 '훈련' 시킬 때 과대적합을 막기 위한 도구일 뿐, 모델의 '순수한 예측 성능'을 평가할 때는 MSE(실제 정답과의 차이)만 보는 것이 타당합니다.

- Step 5:** 최적  $\lambda^*$  선정 (Select Best  $\lambda$ ) 4단계에서 기록한 MSE 값을 중, 가장 낮은 MSE를

기록한  $\lambda$ 를 최종  $\lambda^*$  (람다-스타)로 선정합니다.

6. **Step 6:** (권장) 모델 재훈련 (Refit Model) 최적의 하이퍼파라미터  $\lambda^*$ 를 찾았으므로, 이제 '검증 세트'의 임무는 끝났습니다. 훈련(Train) 데이터와 검증(Validation) 데이터를 다시 하나로 합친 더 큰 데이터셋을 사용하여,  $\lambda^*$  값으로 모델을 단 한 번 재훈련합니다.  
이유: 모델 선택이 끝났으니, 검증에 썼던 데이터도 훈련에 사용하여 최종 모델이 조금이라도 더 많은 정보를 학습하게 합니다.
7. **Step 7:** 최종 평가 (Final Report) 지금까지 단 한 번도 사용하지 않은 테스트(Test) 데이터를 사용하여, 6단계에서 얻은 최종 모델의 성능(MSE)을 평가하고 이 점수를 보고합니다.

## 6.2 방법 2: K-Fold 교차 검증 (Cross-Validation)

단일 검증 세트는 데이터가 어떻게 분할되었느냐에 따라 '운' 좋게 특정  $\lambda$ 에 유리한 결과가 나올 수 있습니다. (e.g., 검증 데이터가 우연히 직선 형태)

\*\*K-Fold 교차 검증(CV)\*\*은 이 '운'의 요소를 제거하여 더 안정적이고 신뢰할 수 있는  $\lambda$ 를 찾는, 더 강력한 방법입니다.

1. **Step 1:** 데이터 분할 (Split Data) 데이터를 (훈련+검증)용 훈련 풀(Training Pool)과 테스트(Test)용으로 2-way 분할합니다.
2. **Step 2:**  $\lambda$  후보군 선정 (Select  $\lambda$  range) (이전과 동일). 예: '[0.001, 0.1, 1, 10, 100]'
3. **Step 3:** K-Fold 분할 (Split K-Folds) 훈련 풀(Training Pool)을 K개 (예: 5개)의 '폴드(fold)'로 균등하게 나눕니다.
4. **Step 4: K-Fold CV 루프 실행 (Run CV Loop)** K번 반복합니다. (예:  $k = 1$ 부터 5까지)
  - **For**  $k = 1$ : Fold 1을 '검증용', Fold 2~5를 '훈련용'으로 사용.
  - **For**  $k = 2$ : Fold 2를 '검증용', Fold 1, 3~5를 '훈련용'으로 사용.
  - ... (K번 반복) ...
 각  $k$ 번째 반복마다, 모든  $\lambda$  후보에 대해 모델을 훈련하고 검증용 폴드의 MSE를 기록합니다.
5. **Step 5: 평균 MSE 계산 (Average MSEs)** 4단계가 끝나면, 각  $\lambda$  후보마다 K개의 MSE 값 (예:  $\lambda = 0.1$  일 때 5개의 MSE)이 쌓입니다. 각  $\lambda$ 별로 K개의 MSE 값의 평균을 계산합니다. (예: ' $\text{AvgMSE}[\lambda = 0.1] = (\text{MSE}_{k1} + \dots + \text{MSE}_{k5})/5$ '')
6. **Step 6: 최적  $\lambda^*$  선정 (Select Best  $\lambda$ )** 5단계에서 계산한 평균 MSE 값을 중, 가장 낮은 평균 MSE를 기록한  $\lambda$ 를 최종  $\lambda^*$ 로 선정합니다.
7. **Step 7: 모델 재훈련 (Refit Model)** K-Fold CV는 오직  $\lambda^*$ 를 찾기 위한 과정이었습니다. 이제 훈련 풀 전체(1~5 Fold 모두)를 사용하여,  $\lambda^*$  값으로 모델을 단 한 번 재훈련합니다.  
이유: K개의 모델 중 하나를 고르는 것이 아니라, 찾은 최적의  $\lambda$ 를 사용하여 \*모든\* 훈련 데이터를 학습한 최종 모델을 얻기 위함입니다.
8. **Step 8: 최종 평가 (Final Report)** 지금까지 단 한 번도 사용하지 않은 테스트(Test) 데이터로 7단계의 최종 모델 성능을 평가하고 보고합니다.

## 7 FAQ 및 주요 질문

**Q:** 럿지(Ridge)와 라쏘(Lasso) 중 무엇을 써야 하나요?

**A:** 정답은 없습니다. 상황에 따라 다르며, 둘 다 시도하고 교차 검증(CV) 점수를 비교하는 것이 가장 좋습니다.

- **라쏘(Lasso)가 유리할 때:** 변수가 수백 수천 개로 매우 많고, 그 중 '소수의 핵심 변수'만 예측에 중요하다고 의심될 때. 라쏘가 불필요한 변수들을 0으로 만들어 변수 선택(Feature Selection)을 자동으로 해줍니다.
- **릿지(Ridge)가 유리할 때:** 모든 변수가 예측에 조금씩이라도 기여한다고 생각될 때. 특히 변수들 간에 강한 상관관계(다중공선성)가 있을 때, 라쏘보다 더 안정적인 성능을 보입니다.

**Q:**  $\lambda$  탐색 범위를 정했는데, 최적 값이 범위의 경계(예: 0.001 또는 100)에서 나왔습니다.

**A:** 탐색 범위를 더 넓혀야 합니다. 만약  $\lambda = 100$ 에서 MSE가 최소였다면, 이는  $\lambda = 1000, 10000$  일 때 MSE가 더 낮아질 가능성이 있다는 신호입니다. 최적의  $\lambda$ 는 U자형 MSE 곡선의 '바닥'에 있어야 합니다. 탐색 범위의 경계에서 최적 값이 나왔다면, 아직 U자 곡선의 바닥을 찾지 못했다는 뜻이므로 범위를 확장하여 다시 시도해야 합니다.

**Q:** (단일 검증) Step 6에서 왜 검증(Validation) 세트를 다시 훈련(Train) 세트에 합쳐서 재훈련하나요?

**A:** 검증 세트의 유일한 임무는 '최적의 하이퍼파라미터( $\lambda^*$ )를 찾는 것'이었습니다. 일단  $\lambda^*$ 를 찾았다면, 검증 세트는 더 이상 필요 없습니다. 이 데이터를 '버리기'보다는, 최종 모델을 훈련시킬 때 훈련 데이터에 다시 합쳐서 모델이 조금이라도 더 많은 데이터를 학습하게 하는 것이 성능에 유리합니다.

단, 테스트(Test) 세트는 절대 훈련에 사용해서는 안 됩니다.

**Q:** K-Fold CV에서 K는 몇으로 정해야 하나요?

**A:** 일반적으로  $K=5$  또는  $K=10$ 을 가장 많이 씁니다.  $K$  역시 하이퍼파라미터지만,  $K$  값을 튜닝하기 위해 또 CV를 하는 것은 비효율적일 수 있습니다. (예:  $K = 5$  일 때와  $K = 10$  일 때의 최종 성능 차이가 미미한 경우가 많음)

$K = 5$  또는  $K = 10$  정도로도 충분히 안정적인  $\lambda$  값을 찾을 수 있습니다.

## 8 1페이지 요약: 빠른 훑어보기

### ▣ 핵심 요약

**핵심 목표:** 일반화 (Generalization) 훈련 데이터가 아닌, '새로운 데이터'를 잘 맞추는 모델을 원한다.

#### 1. 문제: 편향 vs. 분산 (Bias vs. Variance)

- **High Bias (과소적합):** 모델이 너무 단순함. (예측이 정답과 떨어짐)
- **High Variance (과대적합):** 모델이 너무 복잡함. (예측이 데이터마다 널뛰기 함)

#### 2. 진단: 과대적합의 징후

모델의 계수( $\beta$ ) 값이 비정상적으로 크고 불안정해진다. ( $\rightarrow$  "계수 값을 줄여야 한다!")

#### 3. 해결: 정규화 (Regularization)

손실 함수에 '패널티 항'을 추가하여 계수 값이 커지는 것을 억제한다.

$$\text{New Loss} = \text{MSE} + \lambda \times (\text{Penalty})$$

#### 4. 방법: Ridge (L2) vs. Lasso (L1)

- **Ridge (L2)  $\rightarrow \sum \beta_j^2$ :** 계수를 0에 가깝게 줄인다. (안정성  $\uparrow$ )
- **Lasso (L1)  $\rightarrow \sum |\beta_j|$ :** 계수를 정확히 0으로 만든다. (변수 선택 가능)

#### 5. 튜닝: 최적의 $\lambda$ 찾기 (by CV)

1.  $\lambda$  후보 목록을 만든다. (e.g., [0.01, 0.1, 1, 10])
2. 각  $\lambda$ 마다 K-Fold CV를 실행하여 평균 검증 MSE를 계산한다.
3. 평균 검증 MSE가 가장 낮은  $\lambda^*$ 를 선택한다.
4. (훈련+검증) 전체 데이터로  $\lambda^*$ 를 적용하여 최종 모델을 재훈련한다.
5. 테스트(Test) 세트로 최종 성능을 딱 1번 보고한다.