Yonsei University, Department of Applied Statistics

Chapter 5 Resampling Method

YBIGTA Science Team 신보현

February 8, 2019

5. Resampling Methods

Resampling Methods는 통계학에서 빼놓을 수 없는 방법으로 이는 훈련 데이터에서 반복적으로 sample을 추출하고 fitted model에 대해서 더 많은 정보를 얻기 위해 관심있는 모델에 각각의 sample을 refit한다. 이러한 방법을 통해 original training sample만 사용해서 얻기 불가능한 정보를 얻을 수 있다. 해당 챕터에서 cross-validation과 bootstrap에 관해서 배운다. 두 방법 모두 통계적 학습 절차의 적용에 있어서 중요한 역할을 한다. cross-validation은 performance 측정, 또는 적절한 flexibility을 설정(parameter tuning)하기 위해 사용된다. model의 performance를 측정하는 것을 model assessment라고 부르고 적절한 flexibility을 선택하는 것을 model selection이라고 부른다. bootstrap은 보통 parameter를 추정하거나 주어진 통계적 학습의 정확도를 측정하기 위해 사용된다.

5.1 Cross-Validation

Ch2에서 테스트 오차 비율과 훈련 오차 비율의 차이점에 대해서 살펴 보았다. 테스트 오차 비율는 새로운 관측치에 대한 반응변수를 예측할 때 발생하는 오차이다. 테스트 오차는 테스트 데이터가 지정되어 있으면 쉽게 계산할 수 있다. 하지만 이것이 흔한 케이스가 아니다. 반면, 훈련 오차 비율은 훈련 데이터를 사용하여 모델을 적합할 때 발생하는 오차 비율이다. 앞서 살펴 보았듯이, 훈련 오차 비율은 모델의 유연성(flexibility)을 늘려감에 따라서 지속적으로 감소했지만 테스트 오차 비율은 U자 모양을 보였다. 즉, 훈련 오차 비율은 테스트 오차 비율을 과소평가(underestimate)하는 경향이 있으므로 이를 전적으로 모델을 평가하는 지표로 사용하면 안 된다. 테스트 데이터가 없는 상황이 많은데, 훈련 데이터만 주어진 상황에서 테스트 오차 비율을 어떻게 계산할까? 이를 위해 훈련 데이터를 나눔으로써 실제로 구할 수 없는 테스트 오차 비율을 추정하는 개념이 도입된다.

5.1.1 The Validation Set Approach

테스트 오차 비율을 추정하기 위한 방법으로 Validation Set Approach가 있다. 그것은 사용 가능한 관측치를 랜덤하게 훈련 세트, validation set,(또는 hold-out set이라고도 불림)으로 나눈다. 모델은 훈련 세트에 적합되고 그 적합된 모델은 validation set(이하 검증 세트라고 하겠음)에 있는 관측치를 predict하기 위해 사용된다. 거기서 나오는 검증 세트 오차 비율은 테스트 오차 비율에 대한 추정치(estimate)이다. 이렇게 우리가 랜덤하게 sample set를 나누는 과정을 반복하면 테스트 MSE에 대한 다른 추정값을 얻을 것이다. validation set approach는 심플하고 강력하지만 두 가지 단점이 있다.

테스트 오차 비율에 대한 추정치인 검증 세트 오차 비율(validation set error rate)은 훈련 세트과 확인세트에 어떤 관측치가 들어가냐에 따라서 매우 variable할 수 있다. 또한 이는 훈련 세트에 포함된 데이터로만 모델을 적합한다. 통계적 방법이 더 적은 관측치로 훈련되었을 때 더 나쁜 성능을 보이기 때문에 이것은 검증 세트 오차 비율가 아마도 테스트 오차 비율를 과대평가(overestimate)할 수도 있다는 가능성을 보여준다. 다음 subsection에서는 이 두가지 문제를 해결하는 validation set approach의 강화 버전인 cross-validation을 살펴본다.

5.1.2 Leave-One-Out Cross Validation

LOOCV는 validation set approach과 비슷하지만 단점을 해결한다. validation set approach과 마찬가지로 LOOCV는 관측치를 두 부분으로 나눈다. 하지만 비슷한 사이즈로 나누는 대신에 하나의 관측치 (x_1, y_1)가 검증 세트에 사용되고 나머지 관측치 $\{(x_2, y_2, \cdots, (x_n, y_n)\}$가 훈련 세트을 구성하는데 사용된다. 다시 말해 $n-1$ training

관측치에 대해 모델을 적합시키고 y_1가 x_1를 사용해서 예측된다. (x_1, y_1)가 모델을 적합시키는데 사용되지 않았기 때문에 $MSE_1 = (y_1 - \hat{y}_1)^2$는 test error에 대해 근사적인 unbiased estimate를 제공한다. 하지만 MSE_1이 test error에 대해 unbiased하더라도 그것은 하나의 관측치 (x_1, y_1)에서만 계산되었기 대문에 굉장히 변동이 크다.

이러한 과정을 동일하게 반복하는데, (x_2, y_2)를 validation data로 정하고 나머지 $n-1$개의 관측치, $\{(x_1, y_1), (x_3, y_3), \cdots, (x_n, y_n)$를 모델을 적합시키며 $MSE_2 = (y_2 - \hat{y}_2)^2$로 계산한다. 이러한 과정을 n번 반복하여 n개의 squared errors, $MSE_1, \cdots, MSE_n$를 구한다. LOOCV는 test MSE를 이러한 n개의 테스트 에러에 대한 추정치를 평균함으로써 테스트 에러를 추정한다.

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} MSE_i \tag{5.1}$$

LOOCV는 validation set approach에 비해서 몇 가지 장점들이 있다. 먼저, 그것은 훨씬 작은 bias를 가진다. LOOCV에서는 $n-1$개의 훈련 세트를 이용하여 모델을 적합시키고 이러한 과정을 n번 반복한다. 이것은 훈련 세트가 대개 original data set에 절반정도 되는 validation set approach와 대비된다. 결과적으로 LOOCV는 테스트 오차 비율를 과대평가하지 않는 경향을 가진다. 또한 validation set approach에서는 훈련 세트 / 검증 세트로 나누는데, 이 때 발생하는 불확실성(randomness)로 인해서 그에 따라 테스트 오차 비율에 대한 추정치인 검증 세트 오차 비율의 변동성이 상당히 높다. 하지만 LOOCV는 여러번 실행해도 동일한 결과를 보여준다. 애초에 LOOCV는 여러 번 시행해도 n번 나뉘는 훈련 세트와 검증 세트가 동일 할 수밖에 없기 때문이다.
하지만 만약 데이터의 개수가 굉장히 많다면, 즉 n이 굉장히 크다면 LOOCV는 모델을 n번 적합해야하고 이는 시간이 매우 오래 걸릴 것이다. 이에 대한 대안으로, 선형회귀나 다항회귀에서 최소자승법을 사용할 때, 아래와 같은 공식을 사용함으로써 시간을 매우 단축할 수 있다. 즉, LOOCV는 n번의 모델을 적합한 후, 테스트 오차 비율에 대한 추정치를 구할 수 있는데, 단 한 번의 최소자승추정법으로 테스트 오차 비율에 대한 추정치를 구할 수 있는 것이다.

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - h_i} \right)^2 \tag{5.2}$$

여기서 $\hat{y}_i$는 최소자승법으로부터 적합된 값이고 h_i는 98쪽 3.37에 정의된 leverage이다.

5.1.3 k-Fold Cross Validation

LOOCV에 대한 대안으로 나온 것이 k-fold CV이다. 이 방법은 관측치를 랜덤하게 근사적으로 동일한 사이즈를 유지하며 k개의 그룹과 fold로 나눈다. 첫 번째 fold는 validation set으로 여겨지고 나머지 $k-1$ folds에 대해서 적합이 된다. MSE_1는 따로 빼둔 관측치(validation set)에 의해서 계산된다. 이러한 절차가 k 번 반복이 된다. 다른 그룹의 관측치는 validation set으로 여겨진다. 이러한 과정으로부터 test error에 대한 k개의 estimates가 생긴다. $MSE_1, \cdots, MSE_k$ 그리고 k-fold CV estimate을 이 값들을 평균함으로써 얻는다.

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i \tag{5.3}$$

LOOCV는 비용과 시간이 많이 들 수 있지만 cross-validation은 매우 보편적인 방법이고 어느 통계적 학습에 사용될 수 있다. 또한 다음 section에서 살펴보겠지만 bias-variance trade-off 측면에서도 좋은 이점을 포함한다.

cross-validation을 시행할 때, 목표는 주어진 모델이 독립적인 데이터에 얼마나 잘 작동하지 결정하는 것이다. 이

경우에서 테스트 MSE의 추정치가 관심사이다. 하지만 상황에 따라서는 추정된 테스트 MSE 곡선에서의 minimum point의 위치에만 관심이 있을 수 있다. 왜냐하면 많은 모델을 사용하면서 가장 작은 테스트 오차가 나오는 모델이 무엇인지 확인하는 것이 목적일 수 있기 때문이다. 이러한 목적에서 테스트 MSE 곡선에서 최소 점의 위치가 중요하지 추정된 테스트 MSE의 최소값이 실제 테스트 MSE의 최소값과 비슷한지는 관심사가 아니다. 가끔 테스트 MSE를 과소평가(설계한 모델의 성능이 좋다고 오판)할 때도 있지만, 모든 CV curves는 올바른 flexibility의 정도를 알아내는데 근접한다. 다시 말해서 추정된 테스트 MSE 곡선의 최소값을 만드는 flexibility가 실제 테스트 MSE의 최소값을 만드는 flexibility와 어느 정도 일치한다는 것이다. 물론 실제 테스트 MSE를 계산하는 것은 테스트 데이터가 주어지지 않으면 불가능하다. 따라서 아래 그림에서는 데이터를 임의로 생성(simulate)한 후에 실제 테스트 MSE와 10-fold CV를 통해서 추정된 테스트 MSE를 비교해본다.

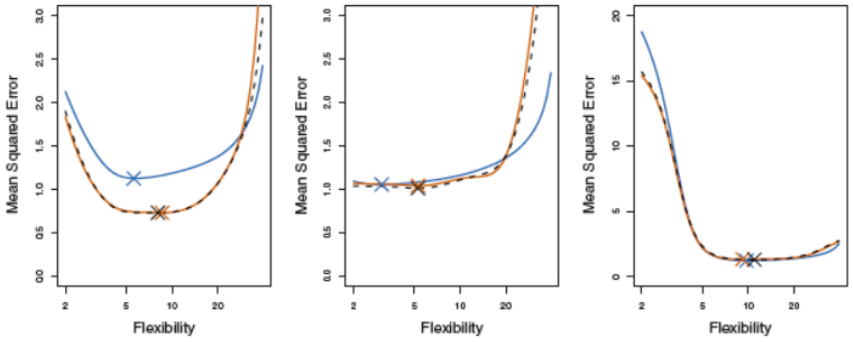
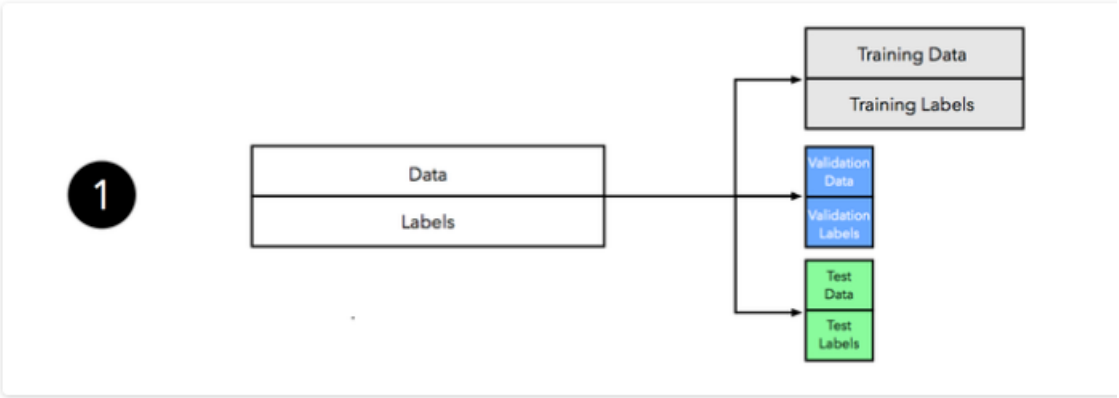


FIGURE 5.6. *True and estimated test MSE for the simulated data sets in Figures 2.9 (left), 2.10 (center), and 2.11 (right). The true test MSE is shown in blue, the LOOCV estimate is shown as a black dashed line, and the 10-fold CV estimate is shown in orange. The crosses indicate the minimum of each of the MSE curves.*

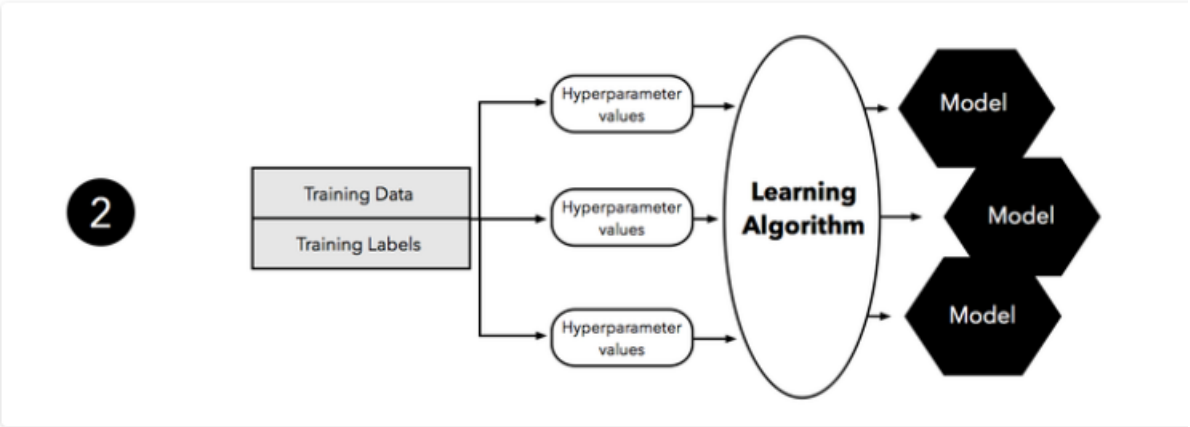
시뮬레이션을 통해서 생성된 세 데이터에 대한 실제 테스트 MSE(파랑색 선), LOOCV의 테스트 MSE 추정치(검정색 점선) 그리고 10-fold CV의 테스트 MSE 추정치(오랜지색 선)이다. 이를 보면 파랑색 선이 최소가 되는 지점이 다른 두 곡선이 최소가 되는 지점이랑 크게 다르지 않음을 확인할 수 있다. 물론, 실제 상황에서는 테스트 MSE를 계산할 수 없기 때문에, LOOCV나 cross-validation을 통해서 계산된 테스트 MSE의 곡선이 최소가 되는 지점을 실제 테스트 MSE 곡선이 최소가 되는 지점과 유사하다고 생각하며 넘어가면 될 듯 하다.

Additional Explanation (3-Hold out Method)

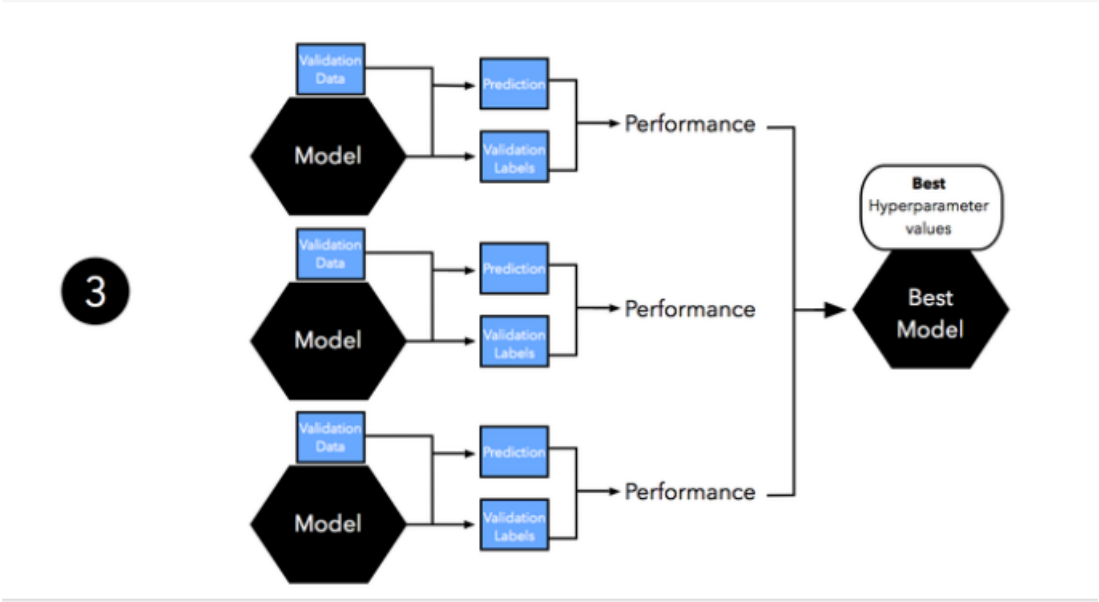
ISLR 책에서는, 데이터를 훈련 세트와 검증 세트로 나누어 모델의 성능을 평가하고 최종 모델을 선택하라고 나와 있다. 하지만 다수의 다른 책에서는 데이터를 훈련 세트 / 검증 세트 / 테스트 세트로 나누라고 말한다. ISLR 책에서는 모델의 하이퍼파라미터를 튜닝하는 작업은 제외하고 설명이 되어 있었다. 하지만 이는 모델을 적합하는데 필수적인 과정이다. 그렇다면 하이퍼파라미터를 튜닝 할 때, ISLR 책에서 나눈 훈련 세트와 검증 세트 중 훈련 세트를 이용하여 여러 파라미터 중 가장 좋은 성능을 보이는 파라미터를 선택하고 다시 동일한 훈련 세트로 모델을 적합하는 것이 올바른 방법일까?
아래 그림에서 구체적인 과정을 살펴보자.



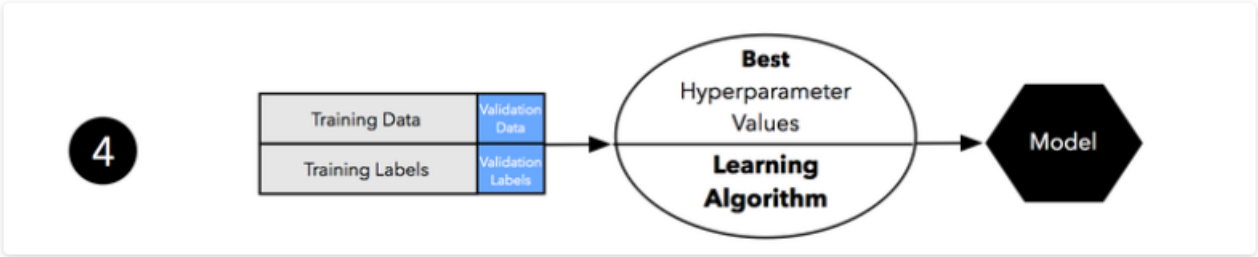
1단계에서는 먼저 데이터를 훈련 데이터 / 검증 데이터 / 테스트 데이터로 나눈다. 훈련 데이터는 모델 학습을 위해, 검증 데이터는 모델 선택을 위해, 테스트 데이터는 최종 선택된 모델을 평가하기 위한 목적이다.



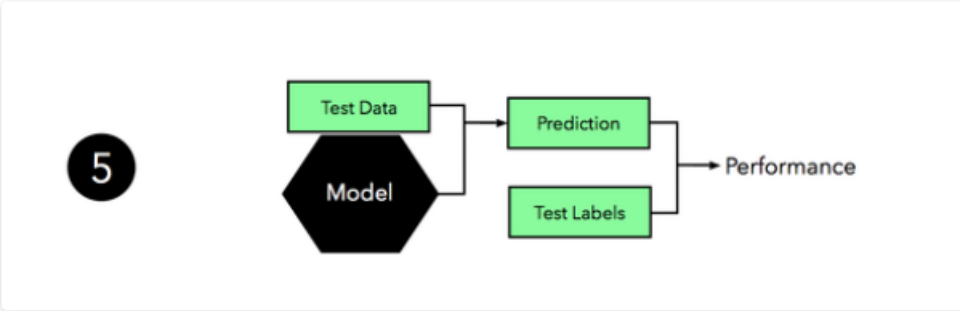
2단계는 하이퍼파라미터 튜닝 단계를 보여준다. 학습 알고리즘, 즉 통계적 학습을 사용하여 여러가지(여기서는 세 가지) 하이퍼파라미터 세팅으로 훈련 데이터에 모델을 학습시킨다.



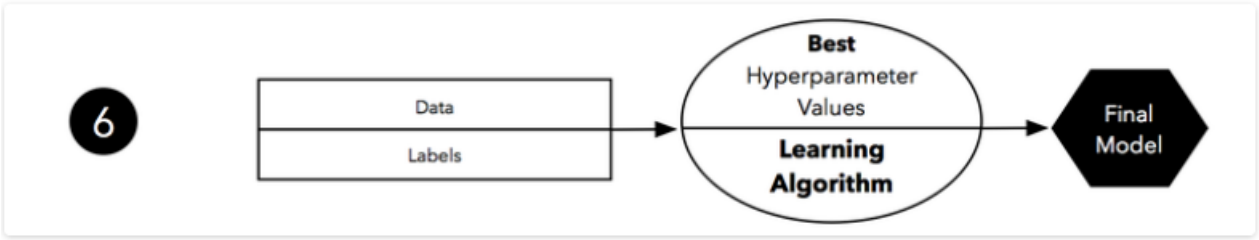
다수의 하이퍼파라미터로 학습시킨 만큼 모델이 생길 것이며 그 모델 중 어느 것이 가장 좋은 모델인지 판단을 해야 한다. 이 때, 훈련 데이터로 다시 판단을 하는 것이 아니라, 해당 모델을 형성할 때 쓰이지 않았던 검증 데이터로 모델의 성능을 평가한다. 그에 따라서 각각 다른 하이퍼파라미터로 생성된 모델 중 가장 뛰어난 모델이 선택받으며 이 때의 하이퍼파라미터도 가장 좋은 하이퍼파라미터라고 생각한다.



여기까지, 가장 좋은 성능을 내는 하이퍼파라미터 값과 모델이 생성되었다. 그런데 훈련 데이터와 검증 데이터를 나눔으로써 데이터 개수가 적어져, 모델의 분산이 클 수도 있다. 그러므로 훈련 데이터와 검증 데이터를 합쳐 더 큰 데이터셋으로 이전 단계에서 얻은 최선의 하이퍼파라미터 설정을 사용하여 모델을 다시 학습시킨다.

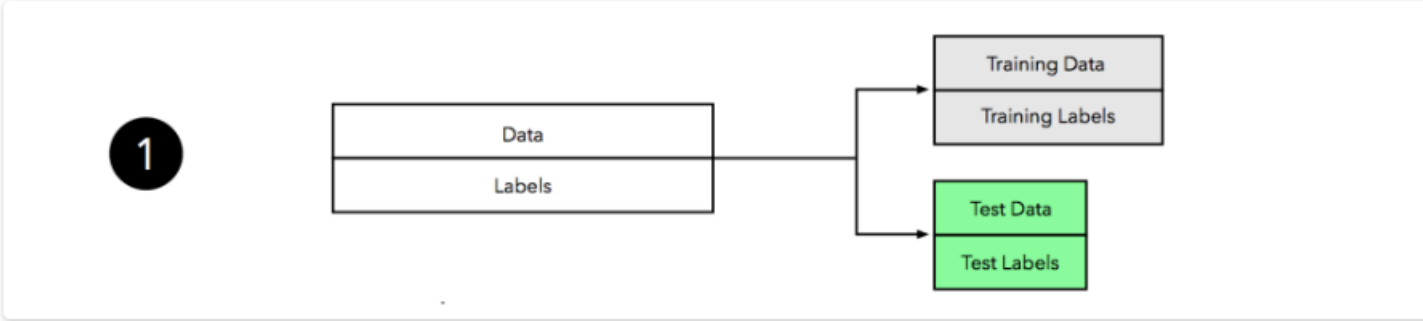


이제 모델의 성능을 평가하기 위해 이제까지 한 번도 사용된 적이 없는 테스트 데이터를 이용한다. 만약 이때, 이전에 사용한 훈련 데이터를 이용하여 모델의 성능을 평가한다면, 이 데이터에 대해서 모델은 이미 학습을 했으므로 과도하게 좋은 결과(과적합)가 나올 것이다.

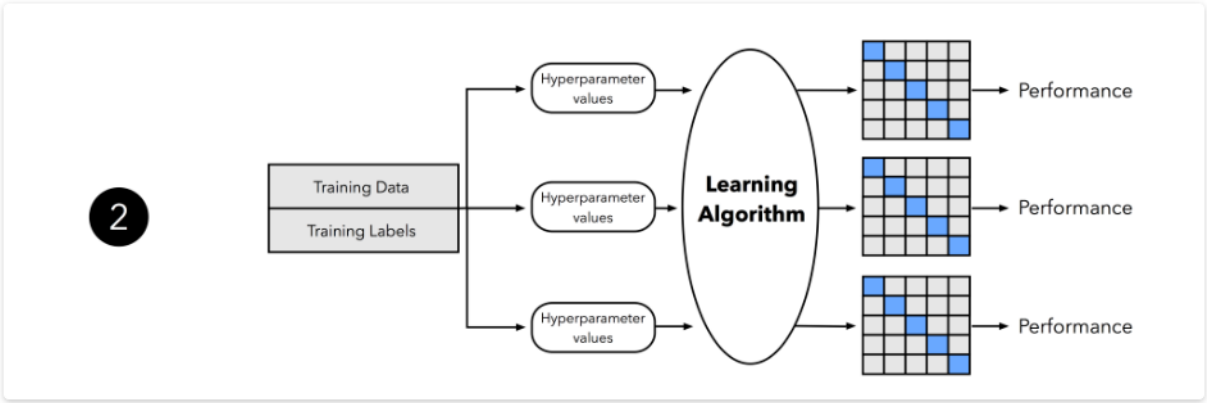


마지막으로 훈련 데이터와 테스트 데이터를 합쳐 전체 데이터를 사용해 실제 운영 환경에 사용할 모델을 학습시킨다.

이상으로 설명한 방법이 3 way hold out method이다. 그런데 이 방법은 데이터를 어떻게 나누느냐에 따라서 성능이 좌지우지될 수 있다. 또한 위의 그림을 보면 데이터를 세 부분으로 나눌 때, 딱히 어떤 기준이 있는 것으로 보이지는 않는다. 이에 대한 대안이 k-fold cross-validation이다. 그림을 통해서 살펴보자.

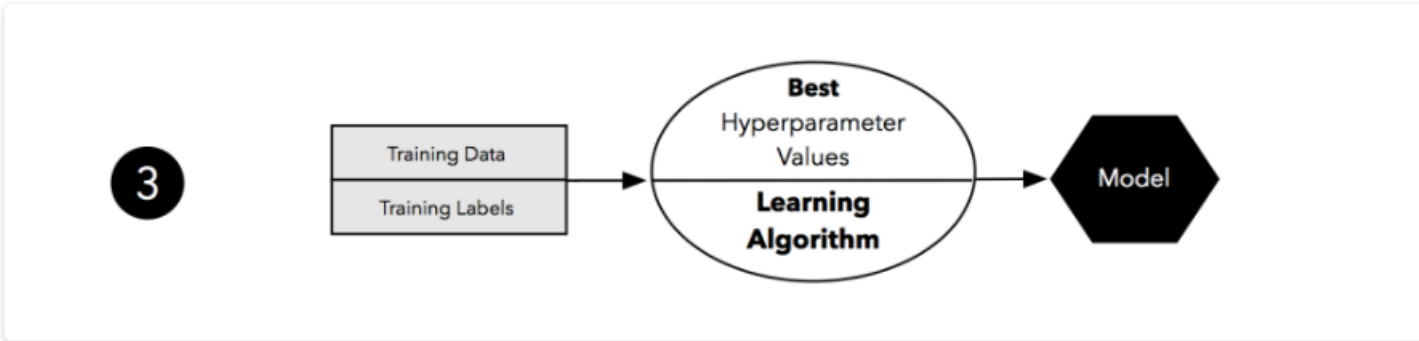


3 way hold out과 마찬가지로 독립적인 테스트 데이터 세트를 구분한다(사실 여기서도 이렇게 훈련 데이터와 테스트 데이터로 나누는 기준이 무엇인지 궁금하다.) 그런데 여기서는 검증 세트를 바로 또 나누지 않는다. 이유는 다음 과정에서 위 그림의 훈련 세트를 다시 훈련 세트와 검증 세트로 나누기 때문이다.

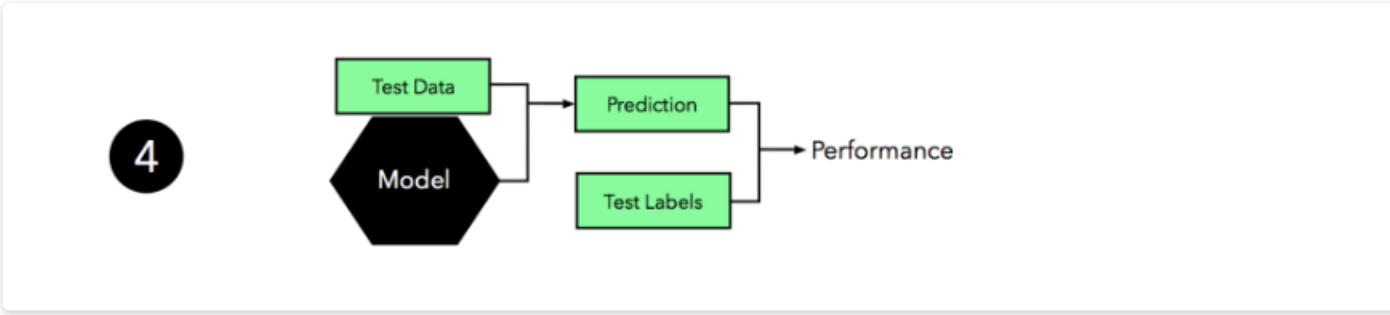


“

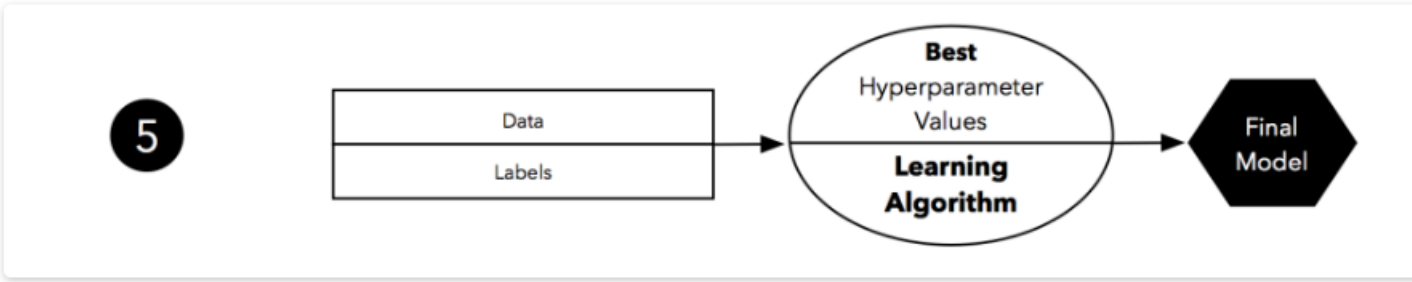
두 번째 단계에서는 여러 하이퍼파라미터 세트를 훈련 세트에 적용해본다. 여기서, 각 하이퍼파라미터 세트를 훈련세트에 적용할 때마다, k-fold cross validation을 적용한다. 즉, 하이퍼파라미터 세트1을 훈련 세트에 적용할 때, k-1개의 겹에 대해서 학습 모델을 적용시키고 나머지 1개의 겹에 대해서 모델 성능 추정치를 구한다. 이러한 과정을 k번 반복하여 평균을 내면 하이퍼파라미터 세트1에 대한 모델 성능 추정치를 구할 수 있다. 이렇게 여러 하이퍼파라미터 세트(위 그림에서는 3 세트)에 대해서 모델 성능 추정치를 구한다.



두 번째 단계에서 가장 좋은 성능을 보이는 하이퍼파라미터 세트를 구한 뒤, 전체 훈련 데이터 세트에 대해서 다시 학습을 시킨다.



이제 맨 처음에 따로 분리한 테스트 데이터 세트를 사용할 차례이다. 세 번째 단계에서 얻은 모델의 성능을 평가하기 위해 여태까지 모델을 학습시킬 때 전혀 사용하지 않았던 독립적인 테스트 데이터 세트를 사용한다.



마지막으로 평가 단계를 완료 했을 때 모든 데이터로 소위 운영 모델이라 부르는 모델을 학습한다.
이와 같이 하이퍼파라미터 튜닝은 모델을 적합하는데 있어서 떼어놓을 수 없는 과정이기 때문에 이를 위해 3-hold out 방법을 사용하는 것이 적절할 것이다. ISLR에서는 이러한 내용이 부족하여 Setabstian Raschka 저, python machine learning을 참조하여 작성했다. 또한 아래의 홈페이지를 참조했다.
www.tensorflow.blog/머신-러닝의-모델-평가와-모델-선택-알고리즘-선택-3/
아래에서 부터는 ISLR의 내용이다.

5.1.4 Bias-Variance Trade-Off for k-Fold Cross-validation

k-fold CV는 LOOCV에 비해서 계산상의 문제가 훨씬 적다. 이러한 자명한 장점 말고도, k-fold CV는 테스트 오차 비율에 대해 더 정확한 추정치를 제공하는데 이는 bias-variance trad-off와 관련이 있다.
section 5.1.1에서 언급되었듯이, validation set approach는 훈련 데이터가 전체 데이터의 반밖에 되지 않기 때문에 테스트 오차 비율를 과대평가할 수 있다. 이러한 논리로는 LOOCV가 거의 전체 데이터의 전부인 $n-1$개의 데이터를 훈련 세트로 사용하기 때문에 테스트 오차에 대해 unbiased estimate를 제공한다고 생각할 수 있다. 그리고 k-fold CV를 시행함으로써 $(k-1)n/k$개(LOOCV보다는 적지만 validation set approach보다는 많은)의

관측치를 포함하기 때문에 중간 정도의 bias를 가질 것이다. 따라서 bias 감소 관점에서는 LOOCV가 k-fold CV 보다 더 선호된다.

하지만 bias가 추정하는 무제에서 단 하나의 걱정거리는 아니다. 반드시 분산도 고려를 해야 한다. 사실, LOOCV는 k-fold CV($k < n$)보다 더 높은 variance를 가진다. 왜 그럴까? LOOCV를 할 때, n개의 적합된 모델의 결과를 평균내는데 각각은 거의 동일한 관측치로 학습이 되었다(데이터 한 개 차이). 그에 따라서 결과물은 서로 매우 양의 방향으로 correlate되어 있다. 이와 반면에, k-fold CV를 할 때, k개의 모델을 적합시키는데 이들은 서로 적합시킨 훈련 세트가 겹치는 정도가 작으므로 서로가 덜 correlate되어 있다. 매우 상관성이 높은 것들의 평균은 그렇지 않은 것들의 평균 더 높은 variance을 가지기 때문에 LOOCV로부터 나오는 테스트 오차 추정치는 k-fold CV보다 높은 분산을 가지는 것이다.

5.1.5 Cross-validation on Classification Problems

여태까지 CV를 Y가 수치형 변수일 때 회귀분석의 관점에서 보았고 MSE를 테스트 오차를 수치화 하는 용도로 사용했다. 하지만 CV는 Y가 범주형 변수일 때 분류 문제에서도 매우 유용하다. 다른 점은 비슷하고 다만 분류 문제에서 CV는 MSE대신 잘못 분류된 관측치의 갯수를 테스트 오차를 수치화하는 용도로 사용한다. 예를 들어, 분류문제에서 LOOCV error rate는 다음과 같은 형태를 취한다.

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} Err_i \tag{5.4}$$

여기서 $Err_i = I(y_i \neq \hat{y}_i)$이다. k-fold CV error rate도 이와 유사하다.

이원 분류 문제에서 로지스틱 회귀를 사용하는 예를 생각해보자. decision boundary을 선형을 할 수도, 차수를 높여서 비 선형으로 할 수도 있다. 이를 어떻게 결정할까? 이러한 결정을 하기 위해서 cross-validation을 사용한다. 아래 그림을 보자.

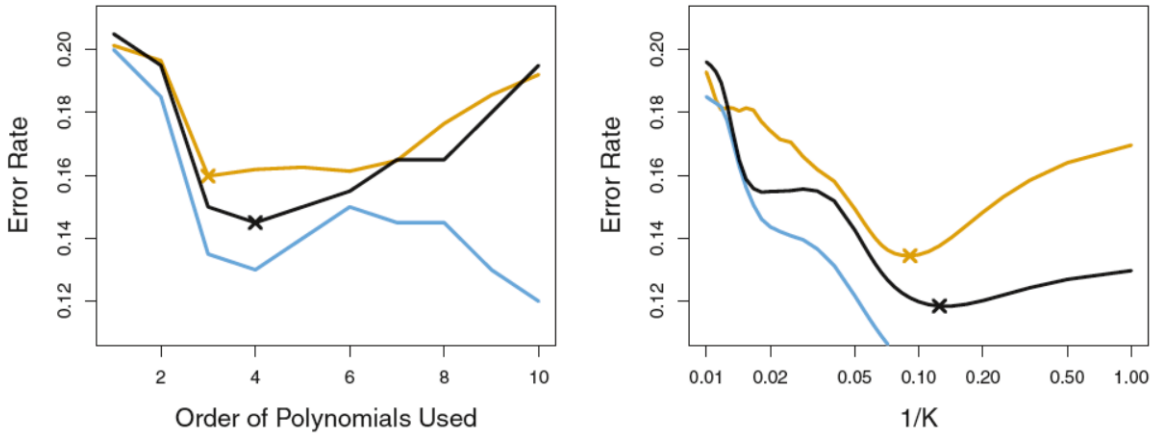


FIGURE 5.8. *Test error (brown), training error (blue), and 10-fold CV error (black) on the two-dimensional classification data displayed in Figure 5.7.* Left: *Logistic regression using polynomial functions of the predictors. The order of the polynomials used is displayed on the x-axis.* Right: *The KNN classifier with different values of K, the number of neighbors used in the KNN classifier.*

갈색 선은 실제 테스트 오차, 파랑색 선은 훈련 오차, 검정색 선은 10-fold CV에 의한 추정된 테스트 오차 곡선이다. 그림을 보면, 전체적으로 갈색 선 아래에 검정색 선이 있음을 확인할 수 있다. 즉 10-fold CV에 의해서 추정된 테스트 오차 비율이 실제 오차 비율을 과소평가(underestimate)하고 있다고 해석할 수 있다. 물론 2장에서 살펴보았듯이, 훈련 오차는 내려가는 경향성을 보인다. 하지만 x로 표시된 지점을 보자. 갈색 선의 x와 검정색 선의 x의 위치는 가로 축에서 크게 차이를 보이지 않는다. 즉, 최소가 되는 지점이 비슷하다는 뜻이다. 따라서 로지스틱 회귀에서 다항식의 차수를 어디까지 정해야하는지, cross-validation을 통해서 추정된 테스트 MSE 곡선이 최소화되는 바로 그 차수(flexibiliry)을 선택하면 된다. 물론 이는 어디까지나 일반화한 결과는 아니다. 하지만 책에서는 일반적인 상황에 대한 증명을 생략했으므로 받아들이면 될 듯 하다.

5.2 The Bootstrap

Bootstrap은 estimator나 통계적 학습 방법에서 불확실성을 수치화하기 위해 사용되는 아주 강력한 툴이다. Bootstrap은 컴퓨터로 구할 수 없는 다양한 것들을 가능하게 해준다. 특히 부트스트랩은 통계적 학습에서 다양하게 적용이 되는데, 통계적인 소프트웨어로 자동적으로 구할 수 없는 변동성을 구하게 해주는 것이 한 예이다.
이번 section에서는 bootstrap을 toy example에 적용하여 살펴본다. X와 Y(얘네들은 random quantities)를 산출하는 재정적인 자산의 고정된 돈의 합계를 투자하고 싶다고 가정하자. X에는 α 의 비율, 그리고 $1-\alpha$만큼 Y에 투자할 것이다. 두 가지 자산의 수익과 관련된 변동성이 있기 때문에, 투자의 total risk 또는 variance를 최소화하는 α를 찾고 싶다. 다시 말해 $Var(\alpha X + (1-\alpha)Y)$를 최소로 하고자 하며 이때 α 값은 아래와 같다.

$$\alpha = \frac{\sigma_Y^2 + \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}} \tag{5.6}$$

여기서 $\sigma_X^2 = Var(X), \sigma_Y^2 = Var(Y), \sigma_{XY} = Cov(X,Y)$ 이다.

현실에서는 이러한 값들이 알려져 있지 않으므로 주어진 데이터를 통해 해당 값을 추정한다.

$$\hat{\alpha} = \frac{\hat{\sigma_Y}^2 - \hat{\sigma}_{XY}}{\hat{\sigma_X}^2 + \hat{\sigma_X}^2 - 2\hat{\sigma}_{XY}} \tag{5.7}$$

이렇게 $\hat{\alpha}$을 구하면 이것이 얼마나 정확한지 수치화해서 확인을 해야, 신뢰할 만한 결과임이 밝혀진다. 즉, 어떤 추정치의 정확성이라 함은, 통계적으로 standard deviation(표준편차)를 의미하며 이를 구하고 싶은데, 이 또한 직접적으로 구하는 것이 아닌 추정을 통해 간접적으로 구한다. 따라서 100 쌍의 (X, Y) 관측치를 생성하고 α을 추정하는 일을 1000번 한다. 따라서 $\hat{\alpha}_1, \cdots, \hat{\alpha}_{1000}$와 같이 1000개의 추정치가 있고 이에 대한 평균값을 구한다($\bar{\alpha}$). 따라서 표준 편차는 아래와 같이 구한다.

$$\sqrt{\frac{1}{1000-1}\sum_{r=1}^{1000}(\hat{\alpha}_r - \bar{\alpha})^2}$$

하지만 실제로는, '원래의 데이터'가 존재하지 않고(즉, 알려진 모수로 모집단이 존재하지 않고) 하나의 sample만 존재하므로 이러한 과정을 할 수가 없다. 하지만 bootstrap은 새로운 sample set를 획득하는 과정을 모방함으로써 추가적인 sample을 만들지 않고 $\hat{\alpha}$의 변동성을 추정할 수 있게 해준다. 다시 말하면, 모집단에서 독립적인 데이터 세트를 반복적으로 뽑는 것이 아니라, 원래의 데이터 세트에서 반복적으로 관측치를 샘플링 한다. 이러한 과정은 아래 그림에서 나타나있다.

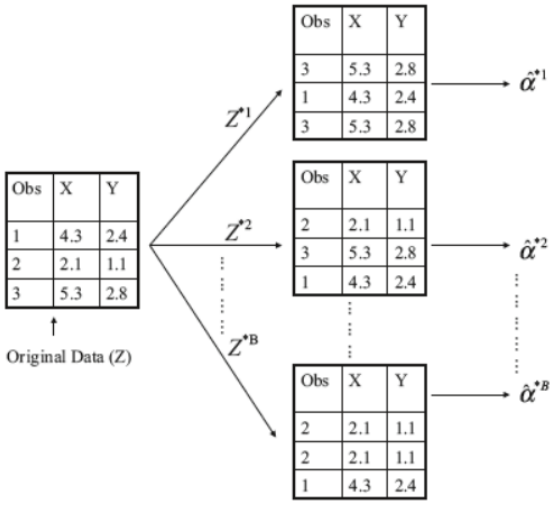


FIGURE 5.11. *A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of α.*

우선 데이터 3개를 포함하는 original data Z를 생각해보자. bootstrap data set, Z^{*1}를 생성하기 위해 랜덤하게 n개의 관측치를 선택한다. 이 과정은 복원 추출로 진행이 된다. 만약 어떤 관측치가 Z^{*1}에 포함이 되어 있다면 X와 Y 값 모두 여기에 포함이 되어있다는 것에 주목하자. Z^{*1}로부터 α를 estimate하기 위한 또 다른 bootstrap을 생성할 수 있다. α에 대한 estimate는 $\hat{\alpha}^{*1}$이라고 표시한다. 이러한 과정이 B개의 다른 bootstrap sample, $Z^{*1}, \cdots, Z^{*B}$를 만들기 위해 B번 반복이 되고 그에 따른 α의 추정치는 $\hat{\alpha}^{*1}, \cdots, \hat{\alpha}^{*B}$이다. 이러한 bootstrap 추정치에 대한

standard error를 다음과 같이 계산한다.

$$SE_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}^{*r} - \frac{1}{B}\sum_{r'=1}^{B}\hat{\alpha}^{*r'}\right)^2} \tag{5.8}$$