
START
머신러닝과 딥러닝
Machine Learning deep Learning

03

Chapter 3. 선형회귀분석 II

Machine Learning & Deep Learning

손영두

e-mail: youngdoo@dongguk.edu

과적합과 과소적합

일반화 (generalization)

- 머신러닝 알고리즘들은 학습 과정을 통하여 점차 학습 데이터에 대하여 오차 (“학습오차”) 를 감소시켜 나감
- 그러나 우리가 학습의 결과로 원하는 모델은 도메인에서 주어진 임의에 데이터에 대하여 성능이 뛰어난 모델, 즉 “일반화 오차(generalization error)”가 작은 모델을 원함
- 따라서 모델의 성능을 평가할 때는 도메인에서 임의로 뽑은, 그리고 학습에 사용되지 않은 데이터들인 테스트 데이터들에서의 성능을 평가해야 함

$$MSE_{test} = \frac{1}{n^{(test)}} \sum_{(x_i, y_i) \in D^{(test)}} (\hat{y}_i - y_i)^2 = \frac{1}{n^{(test)}} \left\| X^{(test)} w - y^{(test)} \right\|^2$$

과적합과 과소적합

과적합

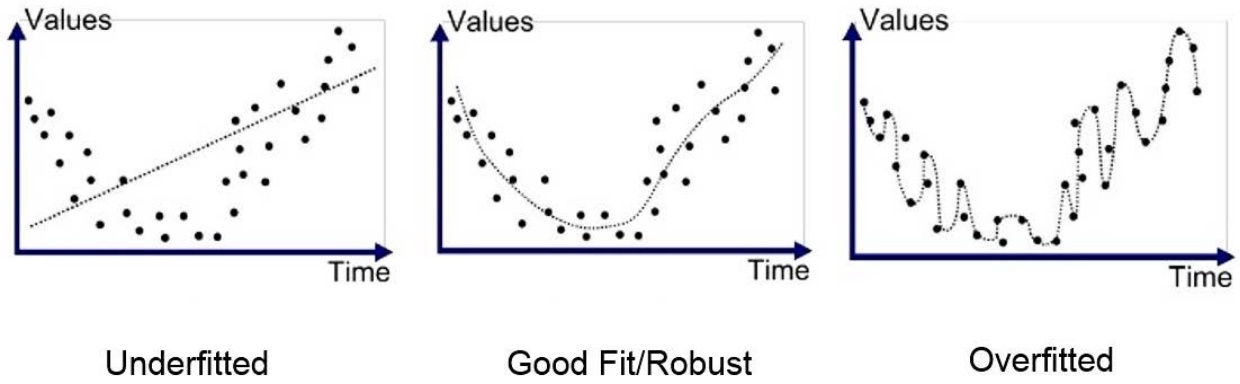
- ✓ 때로는 모델의 학습이 지나치게 학습 데이터에 맞추어져 일반화 성능은 오히려 떨어지는 경우가 존재
- ✓ 이를 "과적합(overfitting)"이라고 함

과소적합

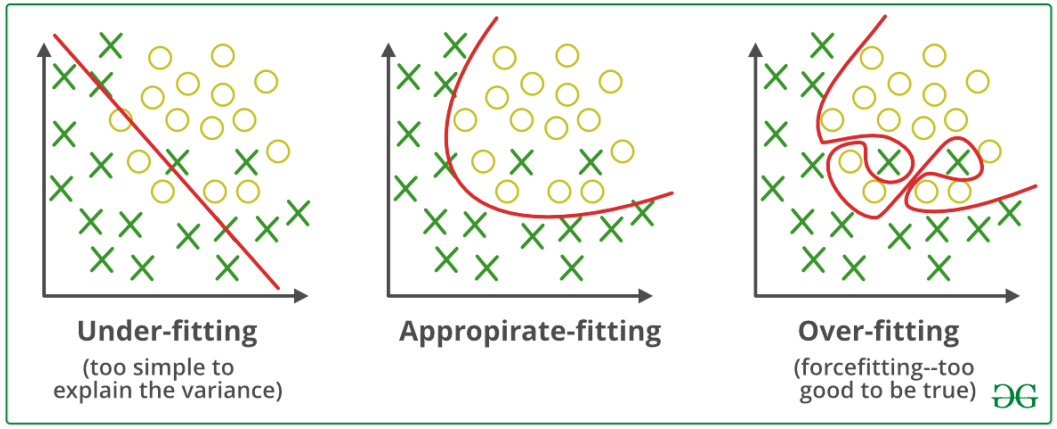
- ✓ 이와 반대로 학습 데이터에 대하여 제대로 학습되지 않아 모델의 성능이 떨어지는 경우도 존재
- ✓ 이를 "과소적합(underfitting)"이라고 함

과적합과 과소적합

회귀분석에서의 과적합 및 과소적합

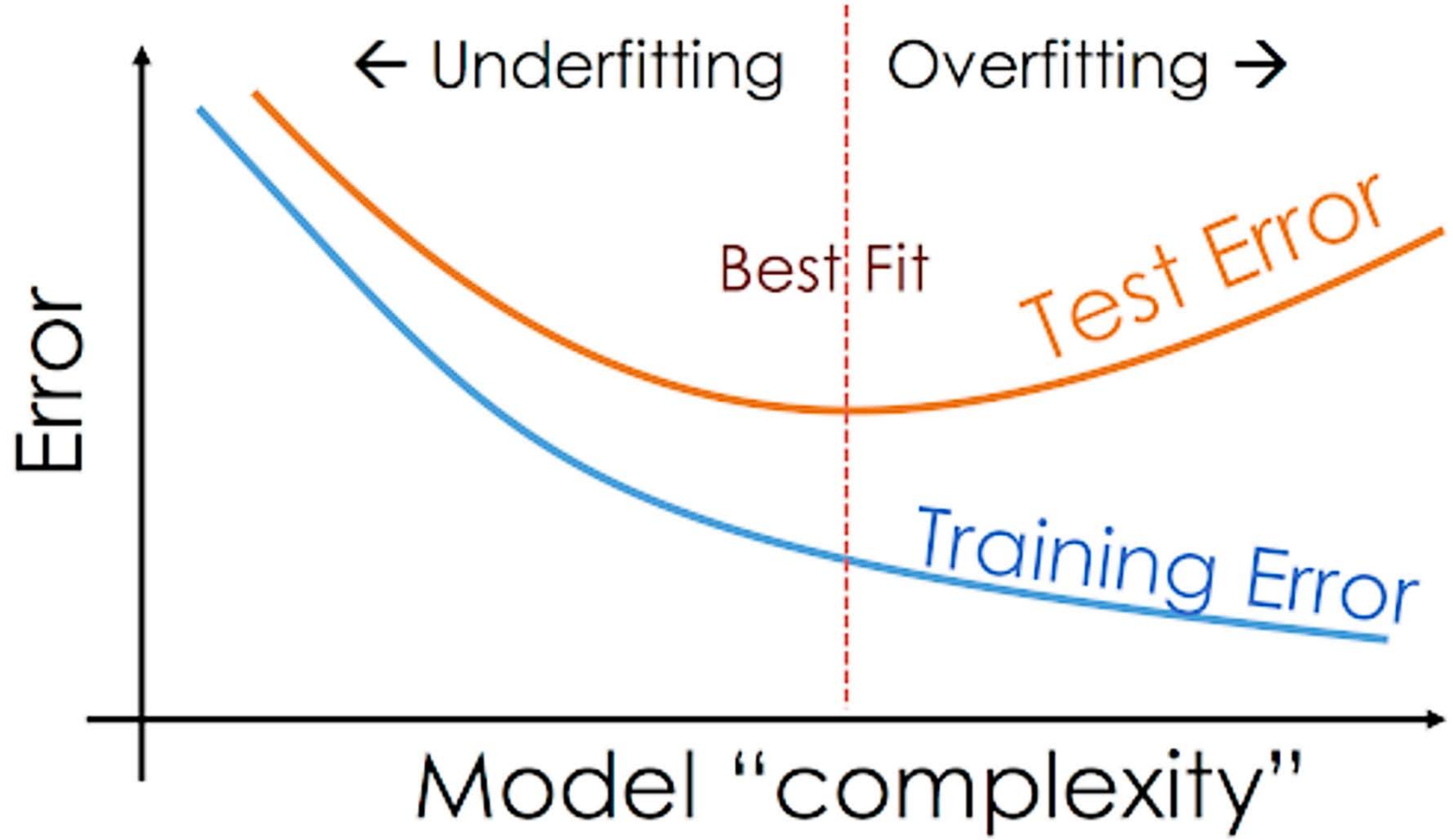


분류에서의 과적합 및 과소적합



과적합과 과소적합

과적합 및 과소적합



선형회귀분석의 과적합

다중회귀분석에서의 과적합

- ✓ 지나치게 많은 독립변수의 사용

곡선회귀분석에서의 과적합

- ✓ 지나치게 높은 차원의 사용

해결 방법

- ✓ 독립변수의 선택 (또는 적절한 다항 차수의 선택)
  - ✓ Forward selection, backward selection, stepwise selection, ….
- ✓ 정규화 (regularization)

REVIEW: 선형회귀분석

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

Minimize

$$\begin{aligned} \mathrm{RSS}(\beta) &= \sum_{i=1}^{N} (y_i - f(x_i))^2 \\ &= \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 \end{aligned}$$

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y},$$

Shrinkage Methods

Ridge regression:

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

또는,

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\text{argmin}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2,$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 \le t,$$

Shrinkage Methods

Ridge regression 회귀 계수:

$$\mathrm{RSS}(\lambda) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^T\beta$$

$$\hat{\beta}^{\mathrm{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

With Singular Value Decomposition (SVD),

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

$$\begin{aligned}
\mathbf{X}\hat{\beta}^{\mathrm{ls}} &= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \\
&= \mathbf{U}\mathbf{U}^T\mathbf{y}, \\
\mathbf{X}\hat{\beta}^{\mathrm{ridge}} &= \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \\
&= \mathbf{U}\,\mathbf{D}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{D}\,\mathbf{U}^T\mathbf{y} \\
&= \sum_{j=1}^{p}\mathbf{u}_j\frac{d_j^2}{d_j^2 + \lambda}\mathbf{u}_j^T\mathbf{y},
\end{aligned}$$

$$\begin{aligned}
\mathrm{df}(\lambda) &= \mathrm{tr}[\mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T], \\
&= \mathrm{tr}(\mathbf{H}_\lambda) \\
&= \sum_{j=1}^{p}\frac{d_j^2}{d_j^2 + \lambda}.
\end{aligned}$$

Shrinkage Methods

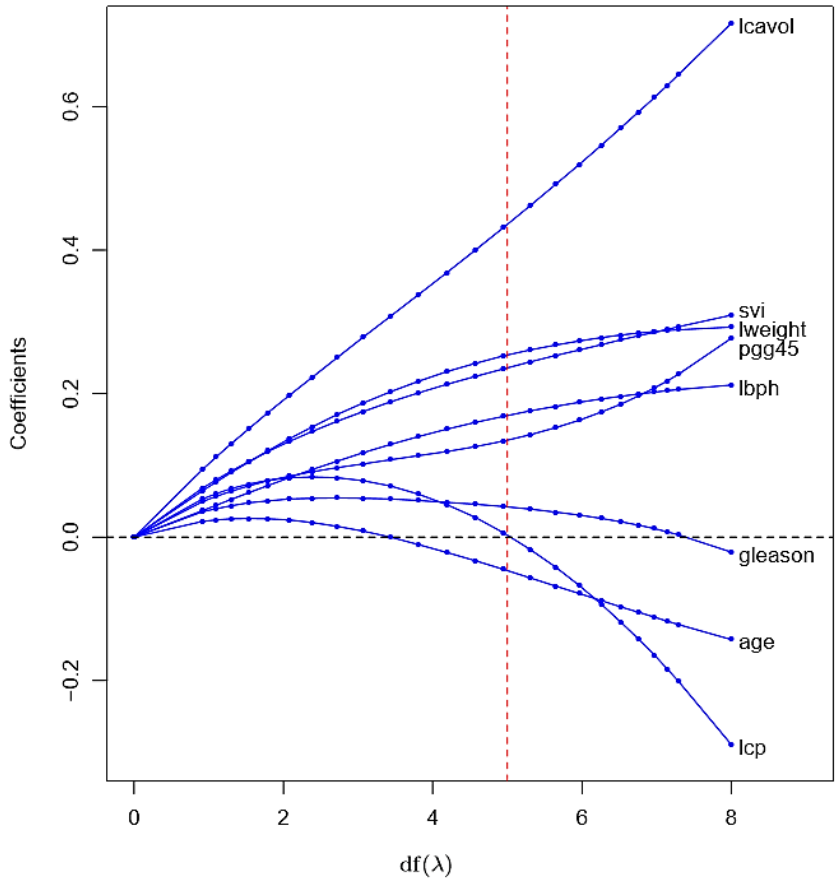


FIGURE 3.8. *Profiles of ridge coefficients for the prostate cancer example, as the tuning parameter λ is varied. Coefficients are plotted versus $\mathrm{df}(\lambda)$, the effective degrees of freedom. A vertical line is drawn at $\mathrm{df} = 5.0$, the value chosen by cross-validation.*

Shrinkage Methods

Lasso regression:

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}.$$

또는,

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2$$

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

Shrinkage Methods

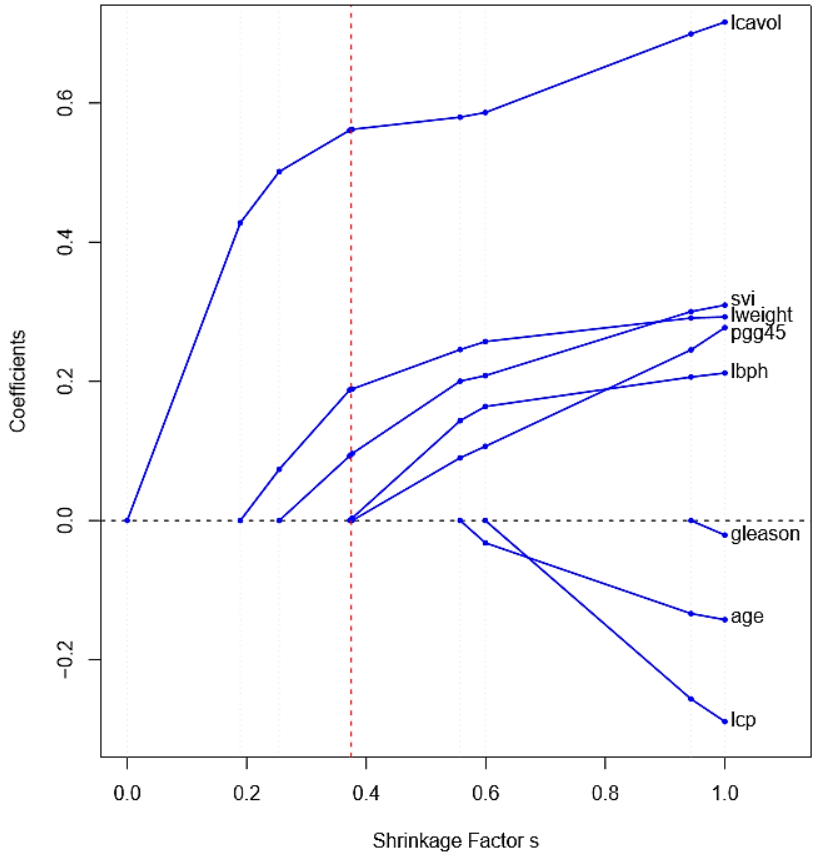
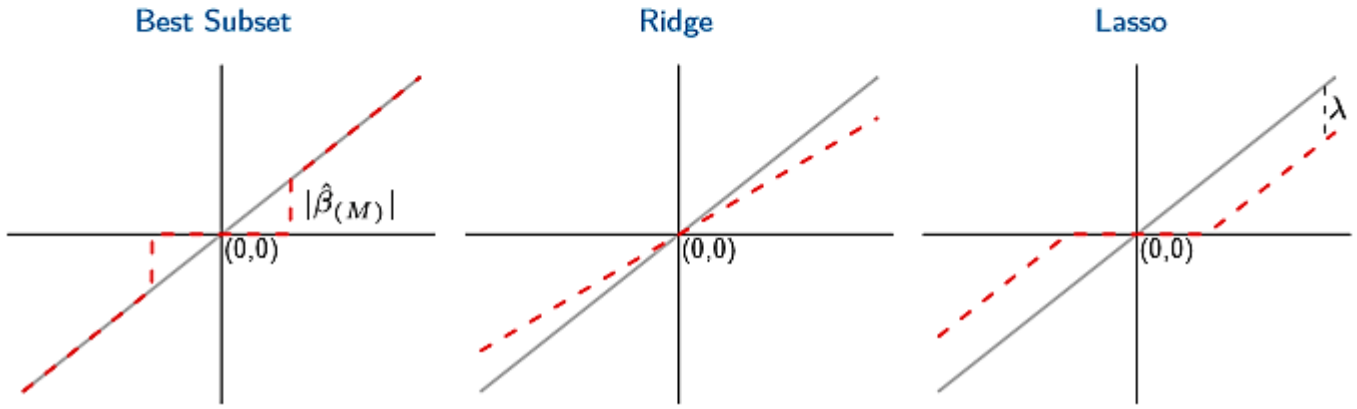


FIGURE 3.10. *Profiles of lasso coefficients, as the tuning parameter t is varied. Coefficients are plotted versus $s = t/\sum_1^p |\hat{\beta}_j|$. A vertical line is drawn at $s = 0.36$, the value chosen by cross-validation. Compare Figure 3.8 on page 65; the lasso profiles hit zero, while those for ridge do not. The profiles are piece-wise linear, and so are computed only at the points displayed; see Section 3.4.4 for details.*

Shrinkage Methods

TABLE 3.4. *Estimators of β_j in the case of orthonormal columns of* $\mathbf{X}$. *M and λ are constants chosen by the corresponding techniques;* sign *denotes the sign of its argument (± 1), and x_+ denotes "positive part" of x. Below the table, estimators are shown by broken red lines. The 45° line in gray shows the unrestricted estimate for reference.*

Estimator	Formula
Best subset (size M)	$\hat{\beta}_j \cdot I(\lvert\hat{\beta}_j\rvert \geq \lvert\hat{\beta}_{(M)}\rvert)$
Ridge	$\hat{\beta}_j/(1+\lambda)$
Lasso	$\text{sign}(\hat{\beta}_j)(\lvert\hat{\beta}_j\rvert - \lambda)_+$



Shrinkage Methods

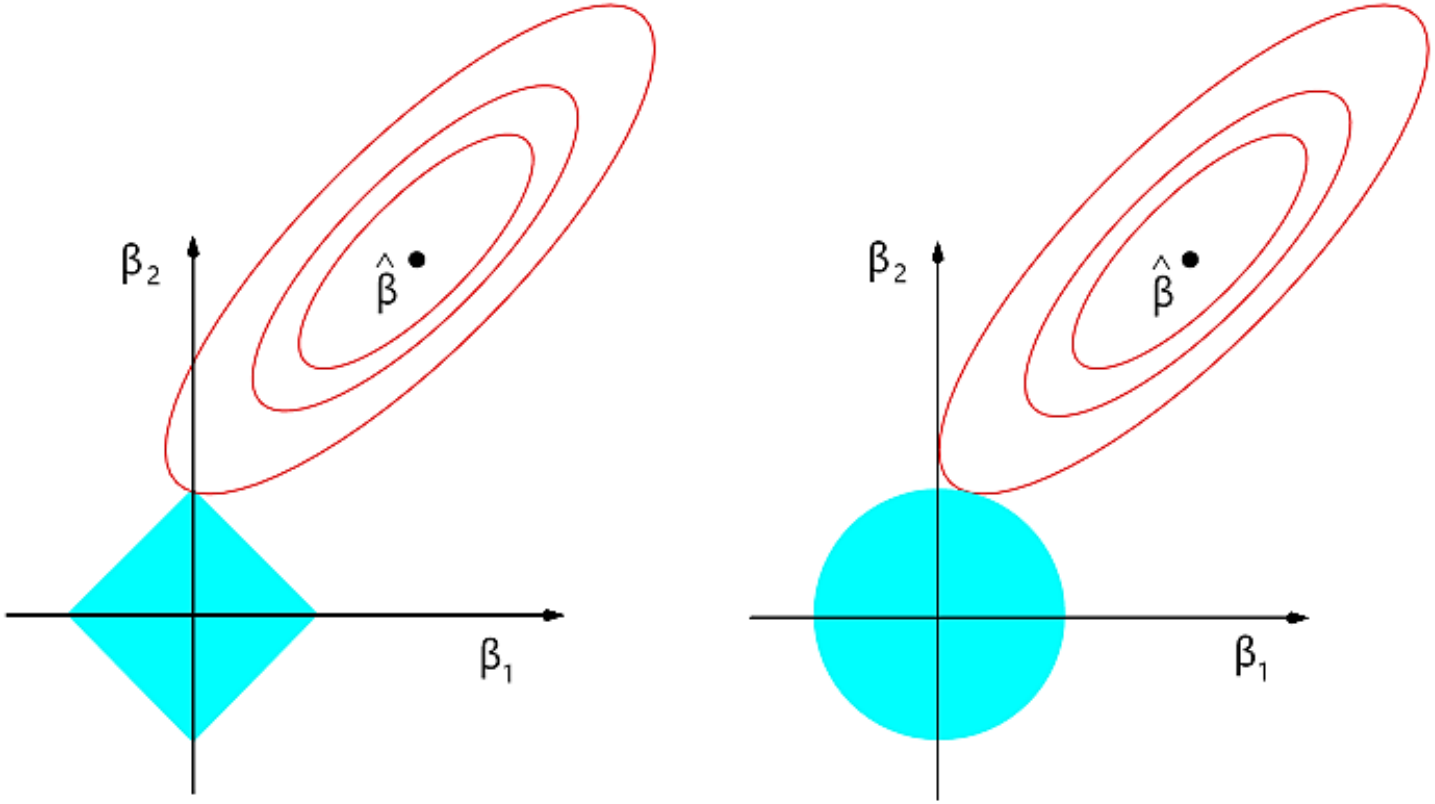


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

Shrinkage Methods

L_q norm regularization

$$\tilde{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|^q \right\}$$

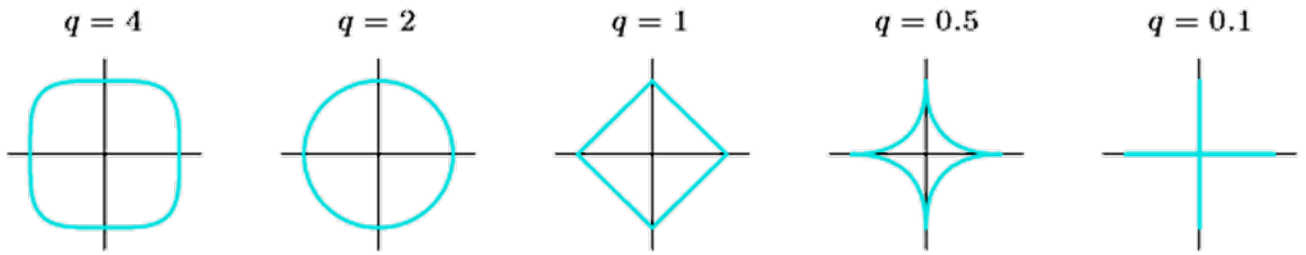


FIGURE 3.12. *Contours of constant value of* $\sum_j |\beta_j|^q$ *for given values of* q.

Elastic net

$$\lambda \sum_{j=1}^{p} \left(\alpha\beta_j^2 + (1-\alpha)|\beta_j|\right).$$

회귀분석의 확률적 해석

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

$$\mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x})\, \mathrm{d}t = y(\mathbf{x}, \mathbf{w})$$

회귀분석의 확률적 해석

우도함수

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1})$$

로그우도함수

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{w}, \beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w})\end{aligned}$$

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2.$$

회귀분석의 확률적 해석

최우도 추정

$$\nabla \ln p(\mathbf{t}|\mathbf{w}, \beta) = \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right\} \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}.$$

$$0 = \sum_{n=1}^{N} t_n \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} - \mathbf{w}^{\mathrm{T}} \left(\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n) \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} \right)$$

$$\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$$

⇨ 최우도 추정을 통해 얻은 최적 회귀계수는 최소제곱법을 통하여 얻게 된 최적 회귀 계수와 같음

(Optional) Bias-Variance Trade-off와 과적합

MSE의 편향-분산 분리

우리가 가지고 있는 데이터가 다음의 알려지지 않은 true function으로부터 생성되었다고 가정하자.

$$y = f^*(\boldsymbol{x}) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2)$$

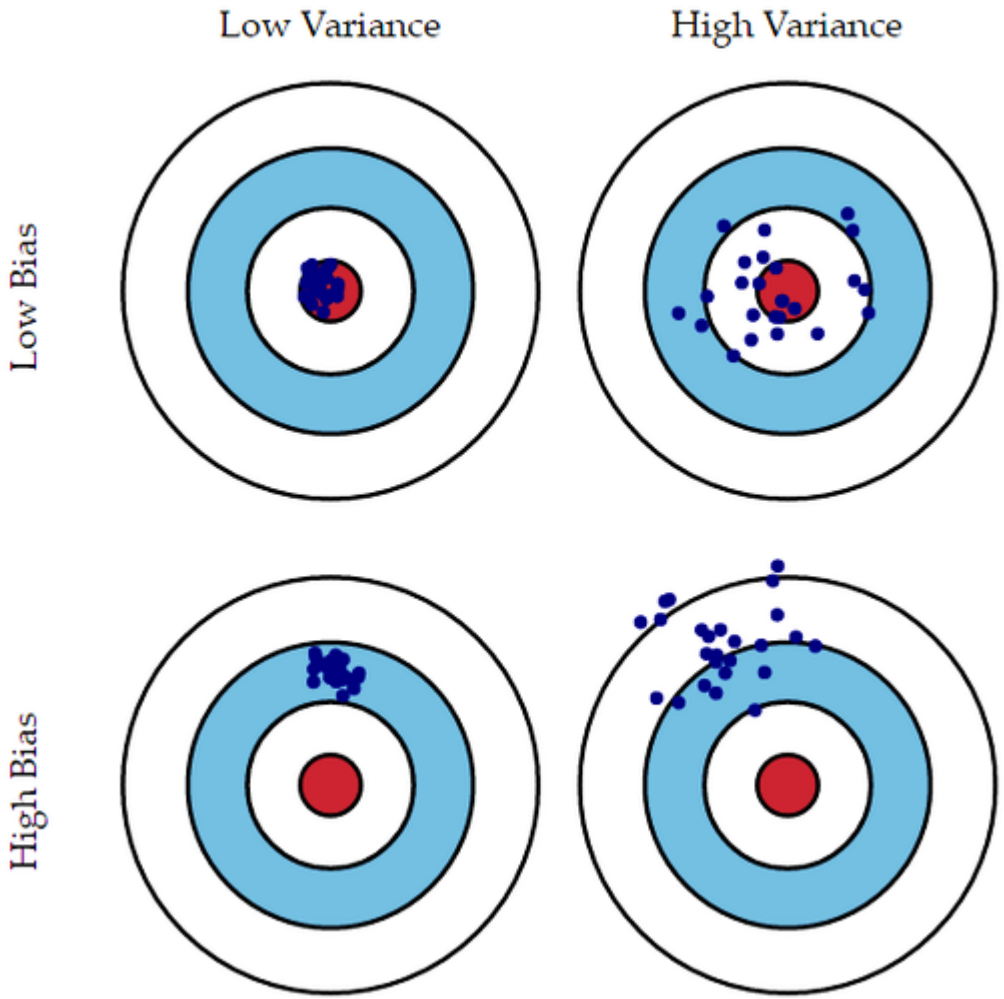
우리의 목적은 학습을 통하여 true function $f^*(\boldsymbol{x})$를 잘 추정하는 좋은 예측 함수 $\hat{y}$ 를 찾는 것이다.

이 때, 일반화 오차의 기대 값은 다음과 같다

$$\mathrm{E}[(y - \hat{y})^2] = (f^*(\boldsymbol{x}) - \mathrm{E}[\hat{y}])^2 + \mathrm{E}[(\hat{y} - \mathrm{E}[\hat{y}])^2] + \sigma^2$$

$$\mathrm{bias}[\hat{y}]^2 = (f^*(\boldsymbol{x}) - \mathrm{E}[\hat{y}])^2$$
$$\mathrm{Var}[\hat{y}] = \mathrm{E}[(\hat{y} - \mathrm{E}[\hat{y}])^2]$$

(Optional) Bias-Variance Decomposition과 과적합



(Optional) Bias-Variance Trade-off와 과적합

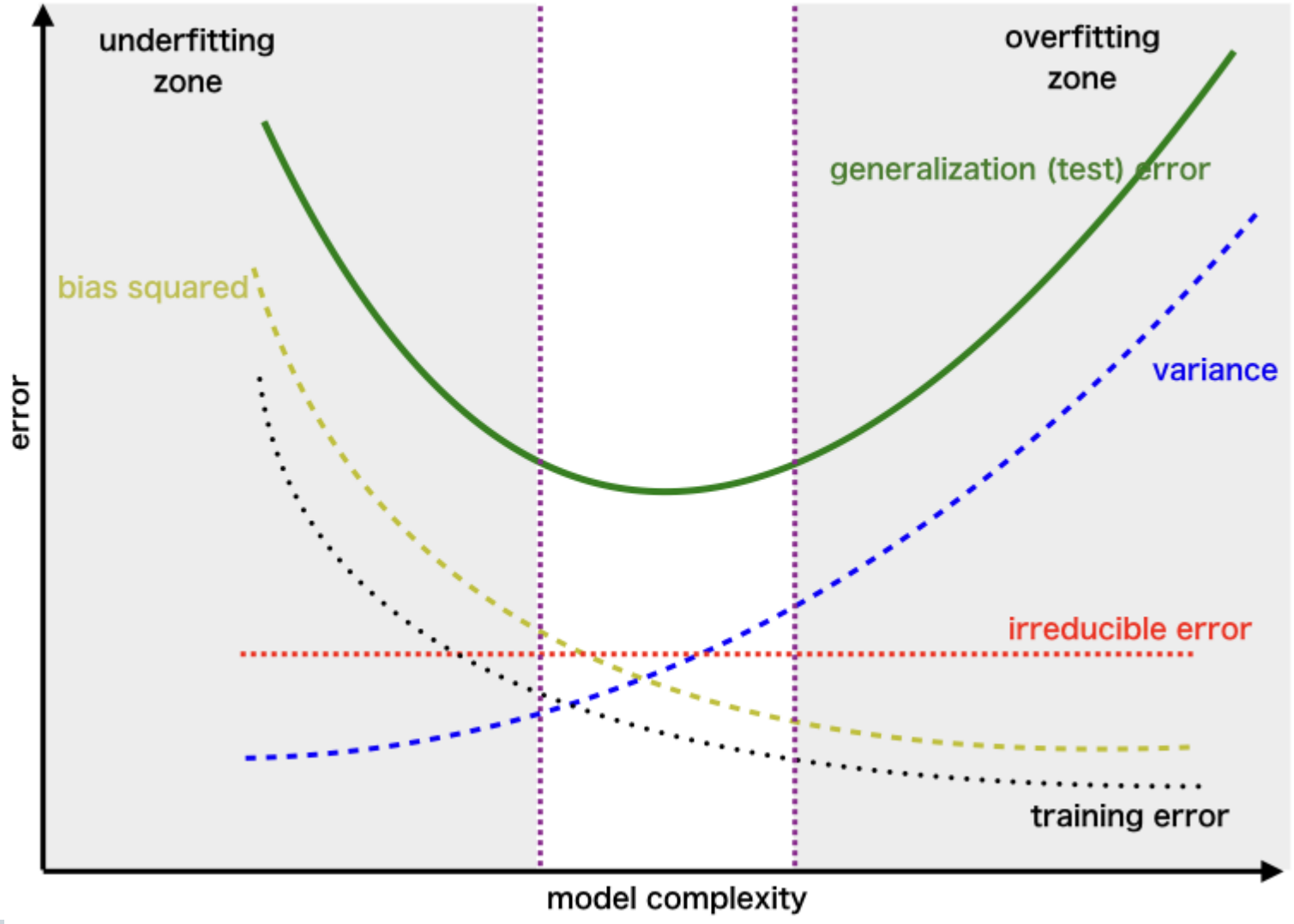
모델의 편향 (Bias)

- ✓ 실제의 종속변수와 모델의 예측 함수의 기대값과의 차이
- ✓ 모델이 학습이 제대로 진행되지 않은 경우 커짐
- ✓ 따라서, 만일 모델의 편향이 지나치게 크다면, 과소적합이 일어났다고 판단
- ✓ 일반적으로 모델의 복잡도가 낮은 경우 편향이 크게 발생

모델의 분산 (Variance)

- ✓ 데이터 샘플에 따른 모델의 예측 함수의 변화의 분산
- ✓ 학습 결과가 지나치게 학습 데이터의 샘플링에 의존적인 경우 커짐
- ✓ 따라서, 만일 모델의 분산이 지나치게 크다면, 과적합이 일어났다고 판단
- ✓ 일반적으로 모델의 복잡도가 높거나 데이터가 적은 경우 크게 발생

(Optional) Bias-Variance Decomposition과 과적합



회귀 모형 만들기 - 데이터 로드

Scikit-learn에서 제공하는 예제 데이터(california housing dataset) load

```
from sklearn.datasets import fetch_california_housing

california = fetch_california_housing()
X=california.data
DF=pd.DataFrame(X,columns=california.feature_names)
Y=california.target
print(DF)
```

회귀 모형 만들기 – 데이터 설명

Scikit-learn에서 제공하는 예제 데이터(California housing dataset)

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85
...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37

	Longitude
0	-122.23
1	-122.22
2	-122.24
3	-122.25
4	-122.25
...	...
20635	-121.09
20636	-121.21
20637	-121.22
20638	-121.32
20639	-121.24

- 타겟 데이터
  - 1990년 캘리포니아의 각 행정 구역 내 주택 가격
- 특징 데이터
  - MedInc : 행정 구역 내 소득의 중앙값
  - HouseAge : 행정 구역 내 주택 연식의 중앙값
  - AveRooms : 평균 방 개수
  - AveBedrooms : 평균 침실 개수
  - Population : 행정 구역 내 인구 수
  - AveOccup : 평균 자가 비율
  - Latitude : 해당 행정 구역의 위도
  - Longitude : 해당 행정 구역의 경도

회귀 모형 만들기 - 학습 및 결과 확인

```
Model=reg.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 4.36693293e-01  9.43577803e-03 -1.07322041e-01  6.45065694e-01
 -3.97638942e-06 -3.78654265e-03 -4.21314378e-01 -4.34513755e-01]
intercept
-36.941920207184 34
```

회귀 계수

Bias

회귀 모형의 정규화 - Ridge(alpha=0.1)

```
from sklearn.linear_model import Ridge
rid=Ridge(alpha=0.1)
Model=rid.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 4.36683387e-01  9.43593980e-03 -1.07303086e-01  6.44965230e-01
 -3.97578456e-06 -3.78652421e-03 -4.21312878e-01 -4.34510858e-01]
intercept
-36.941587163360556
```

회귀 모형의 정규화 – Ridge (alpha=0.5)

```
from sklearn.linear_model import Ridge
rid=Ridge(alpha=0.5)
Model=rid.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 4.36643796e-01  9.43658673e-03 -1.07227325e-01  6.44563694e-01
 -3.97336560e-06 -3.78645054e-03 -4.21306864e-01 -4.34499254e-01]
intercept
-36.940253978928126
```

회귀 모형의 정규화 – Ridge (alpha=1)

```
from sklearn.linear_model import Ridge
rid=Ridge(alpha=1)
Model=rid.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 4.36594382e-01  9.43739513e-03 -1.07132761e-01  6.44062485e-01
 -3.97034295e-06 -3.78635869e-03 -4.21299306e-01 -4.34484717e-01]
intercept
-36.93858523232896
```

회귀 모형의 정규화 – Ridge (alpha=2)

```
from sklearn.linear_model import Ridge
rid=Ridge(alpha=2)
Model=rid.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 4.36495800e-01  9.43901106e-03 -1.06944092e-01  6.43062429e-01
 -3.96430115e-06 -3.78617577e-03 -4.21284056e-01 -4.34455530e-01]
intercept
-36.935240214009 29
```

회귀 모형의 정규화 - Lasso (alpha=0.1)

```
from sklearn.linear_model import Lasso
las=Lasso(alpha=0.1)
Model=las.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 3.90582557e-01  1.50821512e-02 -0.00000000e+00  0.00000000e+00
  1.75019561e-05 -3.32253135e-03 -1.14214430e-01 -9.92250689e-02]
intercept
-7.684589184737931
```

회귀 모형의 정규화 – Lasso (alpha=0.5)

```
from sklearn.linear_model import Lasso
las=Lasso(alpha=0.5)
Model=las.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 2.88854841e-01  1.20314561e-02  0.00000000e+00 -0.00000000e+00
  1.17610340e-05 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00]
intercept
0.5891563081837055
```

회귀 모형의 정규화 – Lasso (alpha=1)

```
from sklearn.linear_model import Lasso
las=Lasso(alpha=1)
Model=las.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 1.45469232e-01  5.81496884e-03  0.00000000e+00 -0.00000000e+00
 -6.37292607e-06 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00]
intercept
1.3480413673416143
```

회귀 모형의 정규화 – Lasso (alpha=2)

```
from sklearn.linear_model import Lasso
las=Lasso(alpha=2)
Model=las.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -2.35579621e-05 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00]
intercept
2.102139496162415
```

회귀 모형의 변수 선택

통계량 계산 패키지 설치(conda install statsmodels)

```
(base) C:₩Users₩seokwon>conda activate machinelearning

(machinelearning) C:₩Users₩seokwon>conda install statsmodels
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

회귀 모형의 변수 선택 – Forward Selection

```
import statsmodels.api as sm
def forward_selection(data, target, cutoff = 0.05):
    initial_features = data.columns.tolist()
    best_features = []
    while (len(initial_features)>0):
        remaining_features = list(set(initial_features)-set(best_features))
        new_pval = pd.Series(index=remaining_features)
        for new_column in remaining_features:
            model = sm.OLS(target, sm.add_constant(data[best_features+[new_column]])).fit()
            new_pval[new_column] = model.pvalues[new_column]
        min_p_value = new_pval.min()
        if(min_p_value<cutoff):
            best_features.append(new_pval.idxmin())
        else:
            break
    return best_features
```

```
forwarddata=forward_selection(DF,Y,0.01)
print(forwarddata)
```

```
<ipython-input-47-48d7b7cbd3dc>:7: DeprecationWarning: The default dtype for empty Ser
e version. Specify a dtype explicitly to silence this warning.
  new_pval = pd.Series(index=remaining_features)
```

```
['MedInc', 'HouseAge', 'Latitude', 'Longitude', 'AveBedrms', 'AveRooms', 'AveOccup']
```

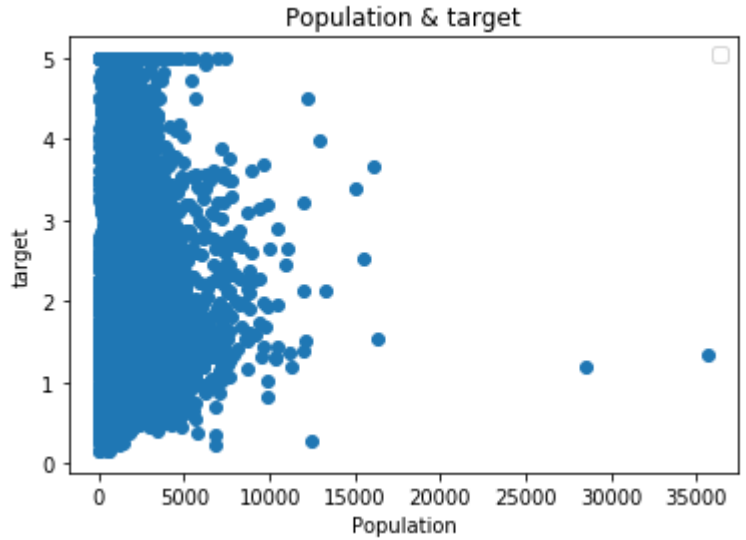
회귀 모형의 변수 선택 – Forward Selection 삭제된 변수

MedInc	행정 구역 내 소득의 중앙값
HouseAge	행정 구역 내 주택 연식의 중앙값
AveRooms	평균 방 개수
AveBedrooms	평균 침실 개수
Population	행정 구역 내 인구 수
AveOccup	평균 자가 비율
Latitude	해당 행정 구역의 위도
Longitude	해당 행정 구역의 경도

$

회귀 모형의 변수 선택 – 삭제된 변수 시각화

```
#i번째 feature와 타겟 값 사이의 관계 시각화
i=4
plt.title(california.feature_names[i]+' & '+ 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.legend()
plt.show()
```



회귀 모형의 변수 선택 – Backward elimination

```
def backward_elimination(data, target,cutoff = 0.05):
    features = data.columns.tolist()
    while(len(features)>0):
        features_with_constant = sm.add_constant(data[features])
        p_values = sm.OLS(target, features_with_constant).fit().pvalues[1:]
        max_p_value = p_values.max()
        if(max_p_value >= cutoff):
            excluded_feature = p_values.idxmax()
            features.remove(excluded_feature)
        else:
            break
    return features
```

```
backwarddata=backward_elimination(DF,Y,0.01)
print(backwarddata)
```

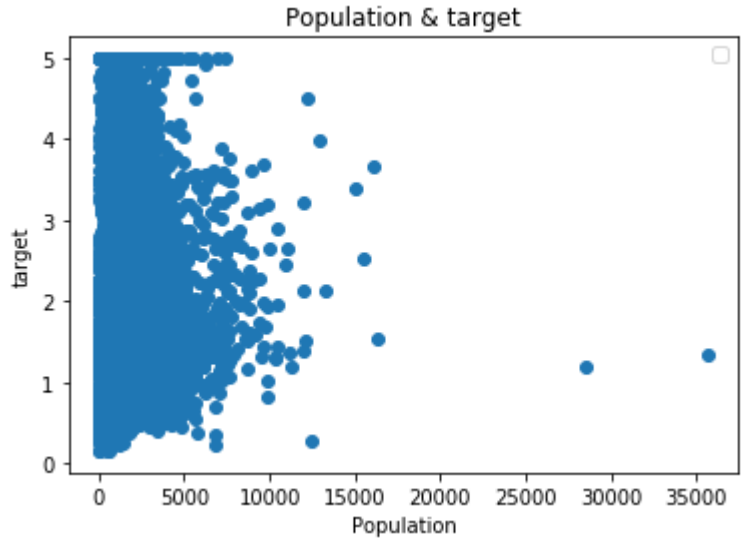
```
['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'AveOccup', 'Latitude', 'Longitude']
```

회귀 모형의 변수 선택 – Backward elimination 삭제된 변수

MedInc	행정 구역 내 소득의 중앙값
HouseAge	행정 구역 내 주택 연식의 중앙값
AveRooms	평균 방 개수
AveBedrooms	평균 침실 개수
Population	행정 구역 내 인구 수
AveOccup	평균 자가 비율
Latitude	해당 행정 구역의 위도
Longitude	해당 행정 구역의 경도

회귀 모형의 변수 선택 – 삭제된 변수 시각화

```
#i번째 feature와 타겟 값 사이의 관계 시각화
i=4
plt.title(california.feature_names[i]+' & '+ 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.legend()
plt.show()
```



회귀 모형의 변수 선택 – Stepwise selection

```
def stepwise_selection(data, target,cutoff):
    initial_features = data.columns.tolist()
    best_features = []
    while (len(initial_features)>0):
        remaining_features = list(set(initial_features)-set(best_features))
        new_pval = pd.Series(index=remaining_features)
        for new_column in remaining_features:
            model = sm.OLS(target, sm.add_constant(data[best_features+[new_column]])).fit()
            new_pval[new_column] = model.pvalues[new_column]
        min_p_value = new_pval.min()
        if(min_p_value<cutoff):
            best_features.append(new_pval.idxmin())
            while(len(best_features)>0):
                best_features_with_constant = sm.add_constant(data[best_features])
                p_values = sm.OLS(target, best_features_with_constant).fit().pvalues[1:]
                max_p_value = p_values.max()
                if(max_p_value >= cutoff):
                    excluded_feature = p_values.idxmax()
                    best_features.remove(excluded_feature)
                else:
                    break
        else:
            break
    return best_features
```

회귀 모형의 변수 선택 – Stepwise selection

```
stepdata=stepwise_selection(DF,Y,0.01)
print(stepdata)
```

```
<ipython-input-55-1d553fc1ac67>:6: DeprecationWarning: The default dtype for empty Se
e version. Specify a dtype explicitly to silence this warning.
  new_pval = pd.Series(index=remaining_features)
```

```
['MedInc', 'HouseAge', 'Latitude', 'Longitude', 'AveBedrms', 'AveRooms', 'AveOccup']
```

회귀 모형의 변수 선택 – Stepwise selection 삭제된 변수

MedInc	행정 구역 내 소득의 중앙값
HouseAge	행정 구역 내 주택 연식의 중앙값
AveRooms	평균 방 개수
AveBedrooms	평균 침실 개수
Population	행정 구역 내 인구 수
AveOccup	평균 자가 비율
Latitude	해당 행정 구역의 위도
Longitude	해당 행정 구역의 경도

$

회귀 모형의 변수 선택 – 삭제된 변수 시각화

```
#i번째 feature와 타겟 값 사이의 관계 시각화
i=4
plt.title(california.feature_names[i]+' & '+ 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.legend()
plt.show()
```