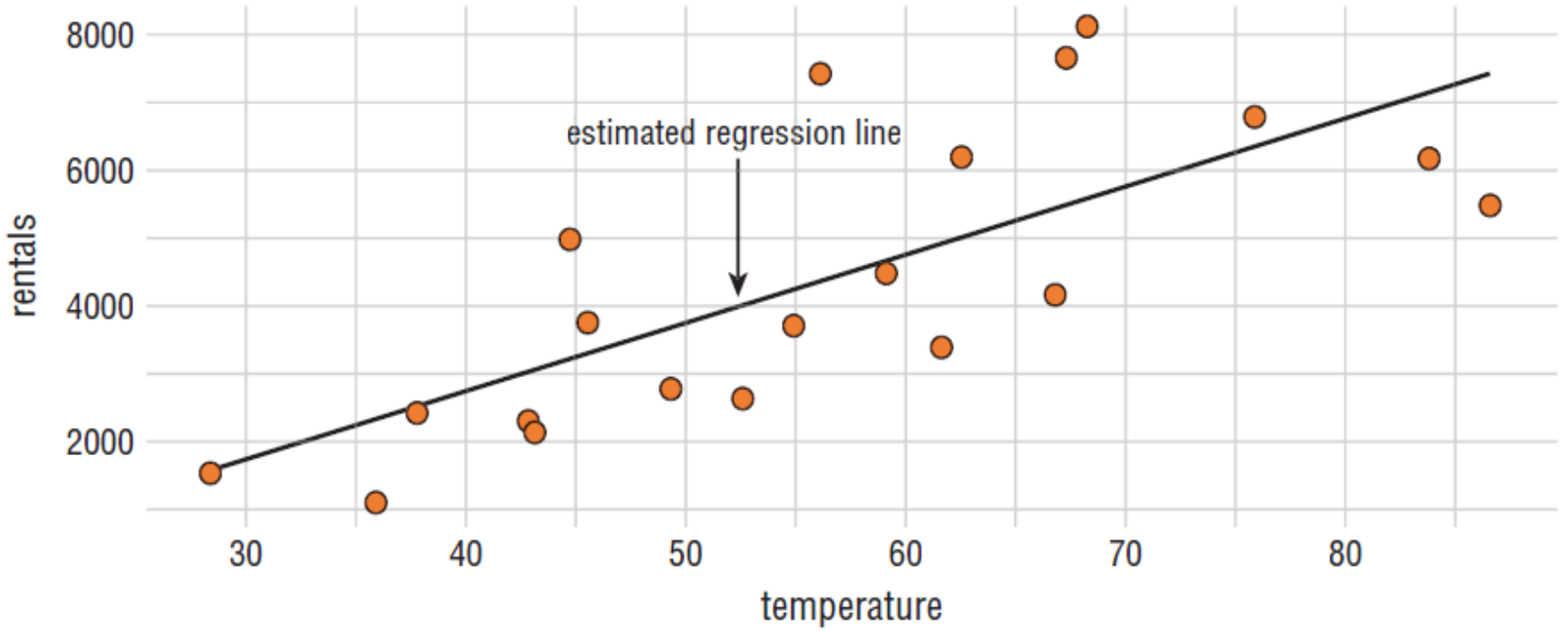
선형회귀분석 기초

선형회귀

□ 기본모델

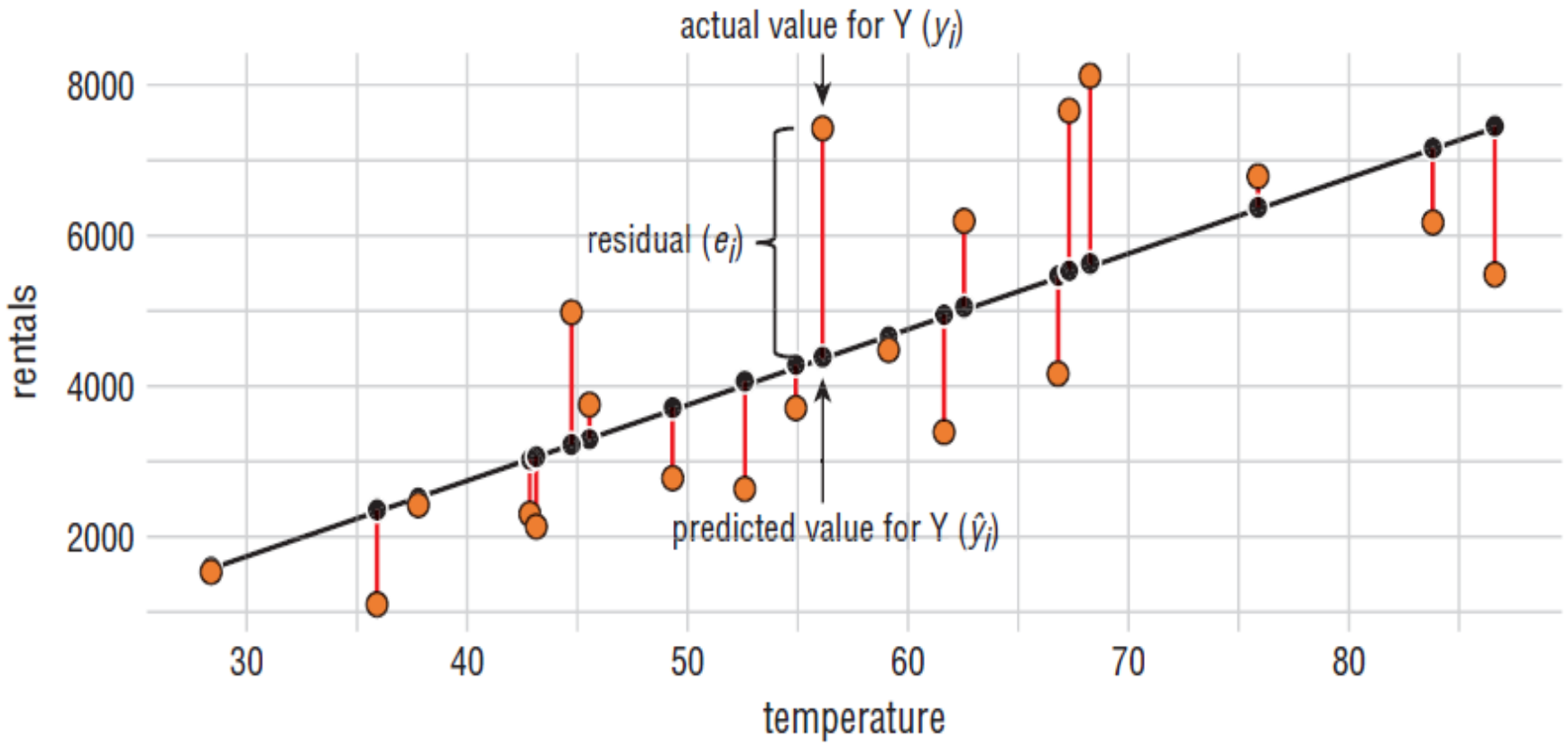
- 한 개의 독립변수(x)와 종속변수(y)의 관계를 잘 설명하는 직선 (회귀선)



선형회귀

□ 기본모델

- 여기서 잔차(오차)는 실제값과 예측값의 차이이며, 이들 제곱의 합을 최소로 하는 선이 바로 원하는 선형회귀선이다..



선형회귀

□ 선형회귀식

$\hat{y} = wx + b$

$\hat{y} = w_0x_0+w_1x_1+w_2x_2+...+w_nx_n+ b$

(또는 $\hat{y} = b_0x_0+b_1x_1+b_2x_2+ ...+b_nx_n+ b_0$로 표현한다.

여기서 w는 계수(가중치) b는 절편(편향) 이다.

y는 종속변수(타겟) x는 독립변수(특성)이다.

y는 참값이고 $\hat{y}$ 는 추정치이고, e는 오차(잔차)다.

□ 비용함수

$$Cost_{lr} = \sum_i (y_i - \hat{y}_i)^2$$

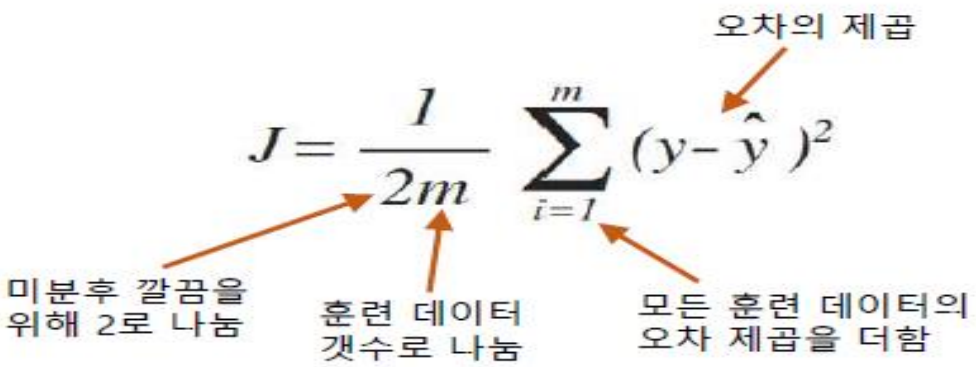
- 참값 y와 모델의 출력값 $\hat{y}$ 의 잔차의 제곱 합을 최소화하는 계수 b와 w를 구하는 것이 목적임

회귀분석의 비용함수

- 회귀분석의 비용 함수 (손실함수, 목적함수, 오차함수)

평균 제곱 오차(Mean Square Error)*

$$J = \frac{1}{2m} \sum_{i=1}^{m} (y - \hat{y})^2$$



해법1: 정규방정식

- 비용 함수 미분

$$\frac{\partial J}{\partial w} = 2 \times \frac{1}{2m} \sum_{i=1}^{m} (y - \hat{y})(-\frac{\partial \hat{y}}{\partial w}) = -\frac{1}{m} \sum_{i=1}^{m} (y - \hat{y}) x$$

$$\frac{\partial J}{\partial b} = 2 \times \frac{1}{2m} \sum_{i=1}^{m} (y - \hat{y})(-\frac{\partial \hat{y}}{\partial b}) = -\frac{1}{m} \sum_{i=1}^{m} (y - \hat{y}) 1$$

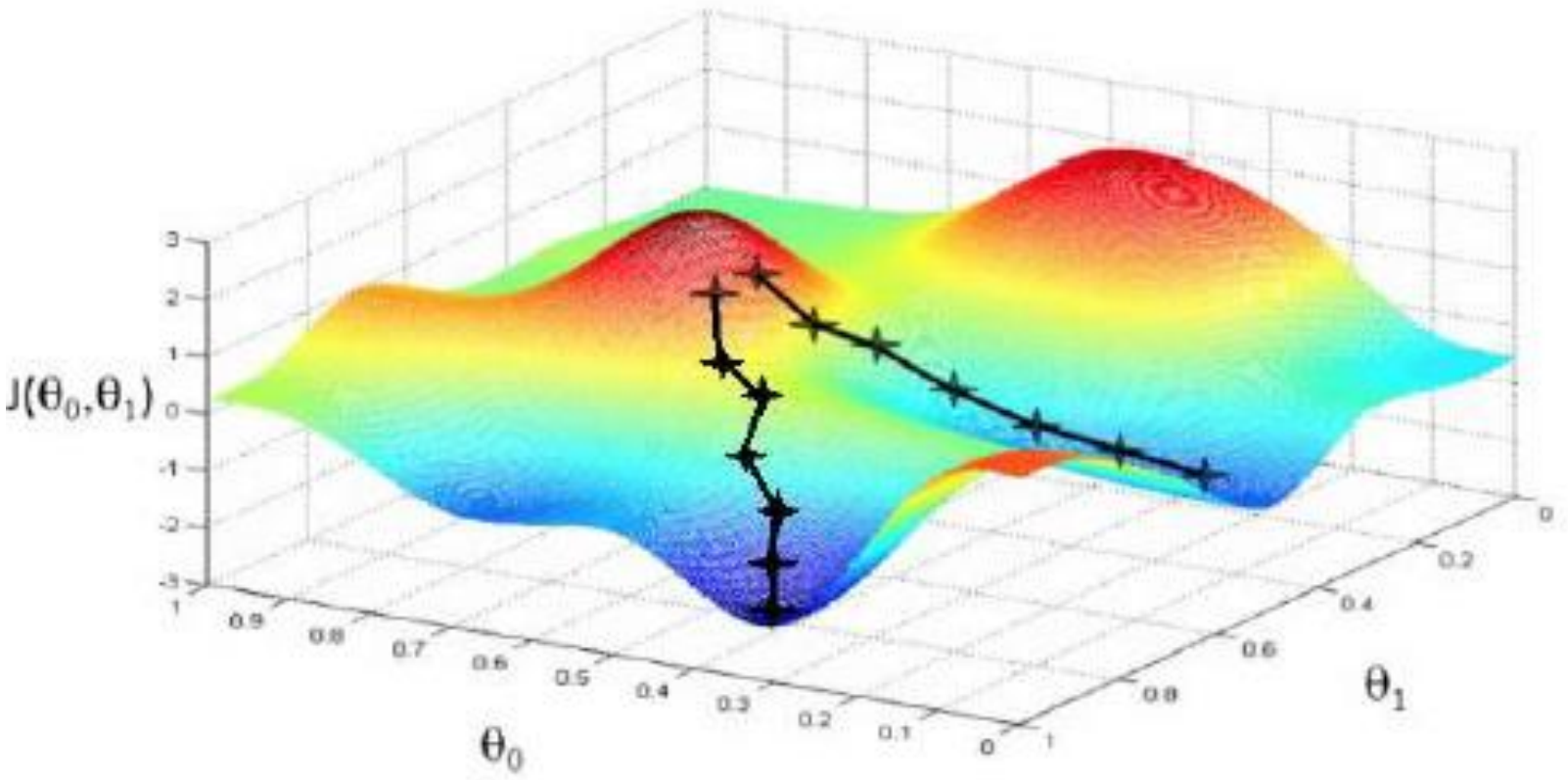
- 정규방정식 (행렬연산을 이용해서 도출)을 이용해서 계수와 편향을 구한다.

$$w = (X^T X)^{-1} X^T y$$

[참고] X의 구성요소간의 상관관계 크면 분포가 0에 가까워지므로, w 추정치의 신뢰도가 떨어진다. (다중공선성)

해법2: 그래디언트 하강법

□ 그래디언트 하강법 (경사하강법)

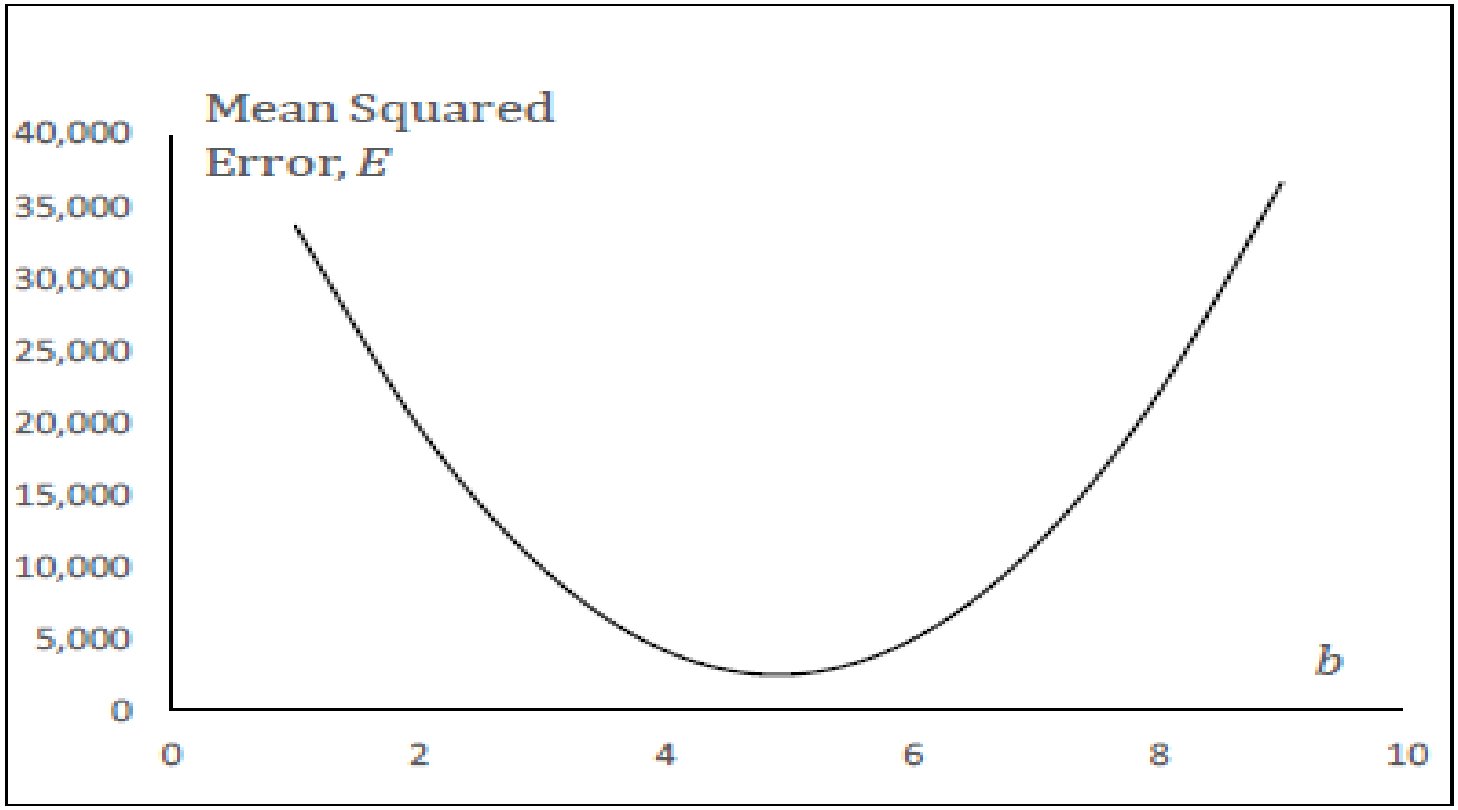


그래디언트 하강법 예시

□ mse과 같은 목적함수를 최소화하기 위해, 그래디언트 하강법은 경사가 가장 가파른 방향을 계산하고, 한 스텝을 취하고 가장 경사가 가파른 방향을 다시 계산하고, 다음 스텝을 취하는 이런 과정을 반복한다.

- 스텝의 크기는 학습률이라고 한다. 스텝이 너무 작으면 알고리즘이 너무 느리며, 너무 크면, 진동할 가능성이 생긴다.

□ 간단한 예: 단순 회귀계수 b의 계산



회귀모델 평가

회귀모델 예측력: RMSE

- 회귀모델 예측력 평가

- RMSE (Root Mean Squared Error) : 평균제곱오차의 제곱근

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

→ 예측모델에서 가장 많이 쓰이는 지표이며, 예측이 대략 평균적으로 RMSE 만큼 오차가 난다고 해석한다.

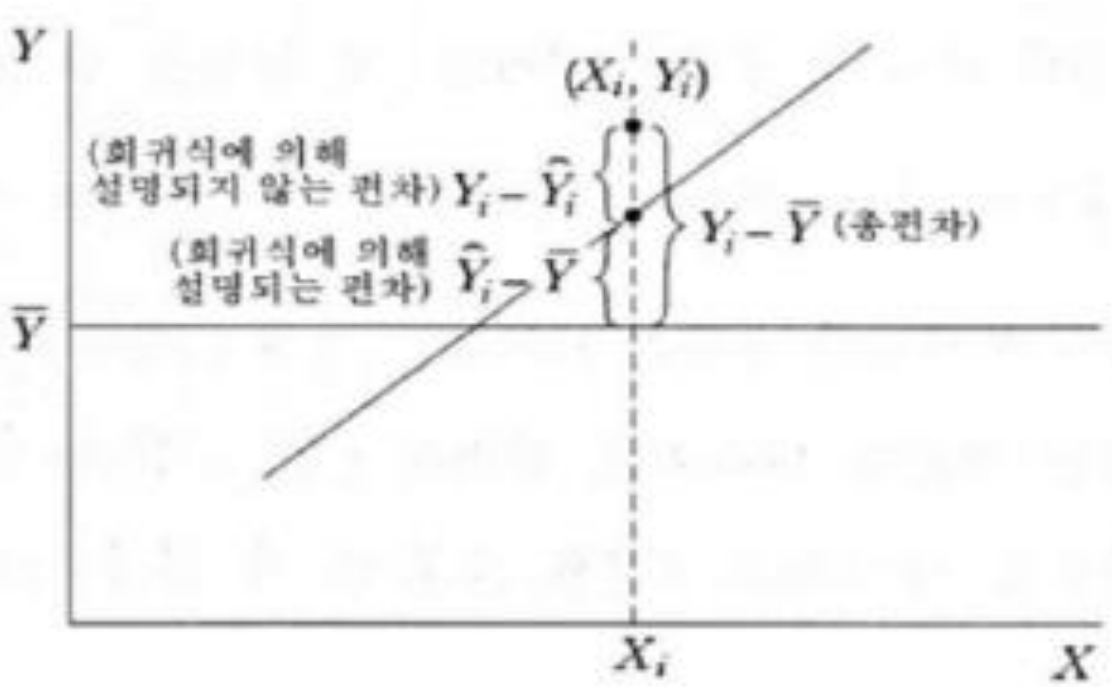
- MAPE(Mean Absolute Percentage Error) :평균절대 백분율오차

$$: \frac{100}{n} \times \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

회귀모델 설명력: R제곱

□ R제곱

- R^2: 결정계수 또는 설명계수라고 하는데 이는 독립변수가 얼마나 타겟변수를 잘 설명하는가를 나타낸다. 0과 1사이에 있으며, 1에 가까울수록 설명을 잘한다고 해석한다.



$$SST = \sum_{i=1}^{n}(Y_i - \bar{Y})^2 \quad SSR = \sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2 \quad SSE = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

SST=SSE+SSR

R^2 = 1- SSE/SST=SSR/SST

=(회귀식에 의해 설명되는 편차)/(총편차)

□ R제곱의 의미

- R제곱 =1 : 모델의 데이터를 환전하게 설명함
- R제곱 =0 : 모델의 데이터를 전혀 설명하지 못함.

기초 실습

선형회귀

□ 단순 선형회귀

모델 불러오기와 정의

```
from sklearn.linear_model import LinearRegression
sim_lr = LinearRegression()
```

모델 적합화

```
sim_lr.fit(x_train['RM'].values.reshape((-1, 1)), y_train)
```

모델 예측

```
y_pred = sim_lr.predict(x_test['RM'].values.reshape((-1, 1)))
```

결과 평가

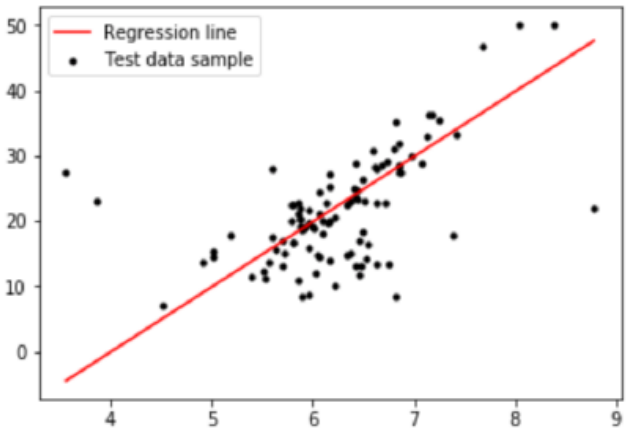
```
from sklearn.metrics import r2_score
```

```
print('단순 선형 회귀, R2 : {:.4f}'.format(r2_score(y_test, y_pred)))
```

```
단순 선형 회귀, R2 : 0.1795
```

```
print('단순 선형 회귀, 계수(w) : {:.4f}, 절편(b) : {:.4f}'.format(sim_lr.coef_[0], sim_lr.intercept_))
```

```
단순 선형 회귀, 계수(w) : 9.9900, 절편(b) : -40.0941
```



선형회귀

□ 다중 선형회귀

```
from sklearn.linear_model import LinearRegression
mul_lr = LinearRegression()
```

```
mul_lr.fit(x_train.values, y_train)
```

```
y_pred = mul_lr.predict(x_test.values)
```

```
print('다중 선형 회귀, R2 : {:.4f}'.format(r2_score(y_test, y_pred)))
```

```
다중 선형 회귀, R2 : 0.6174
```

```
print('다중 선형 회귀, 계수(w) : {}, 절편(b) : {:.4f}'.format(mul_lr.coef_, mul_lr.intercept_))
```

```
다중 선형 회귀, 계수(w) : [-1.39521123e-01  4.17817156e-02 -4.57312740e-03  3.78506106e+00
 -1.46255552e+01  4.52548061e+00  1.49683102e-04 -1.38217694e+00
  2.78132923e-01 -1.03183306e-02 -8.42539713e-01  1.05460752e-02
 -5.19900681e-01], 절편(b) : 27.2753
```

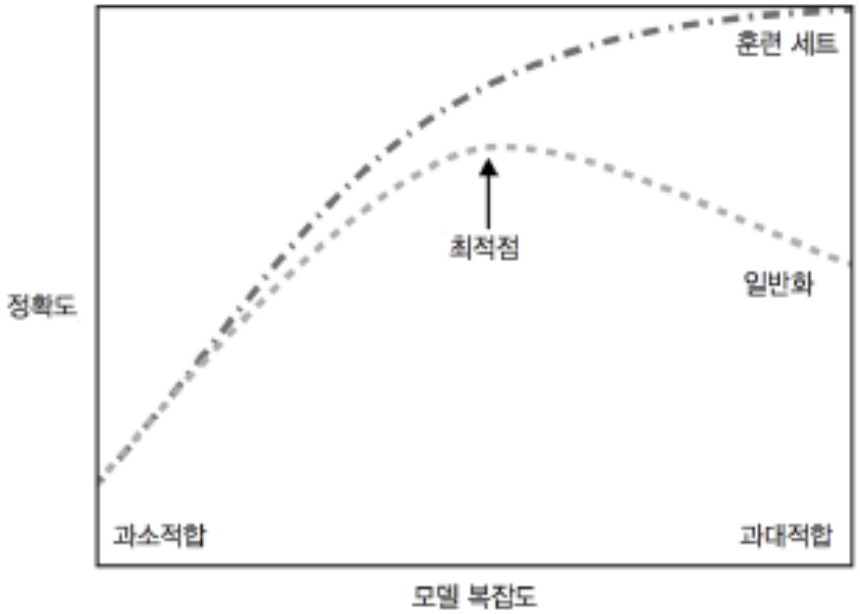
규제화

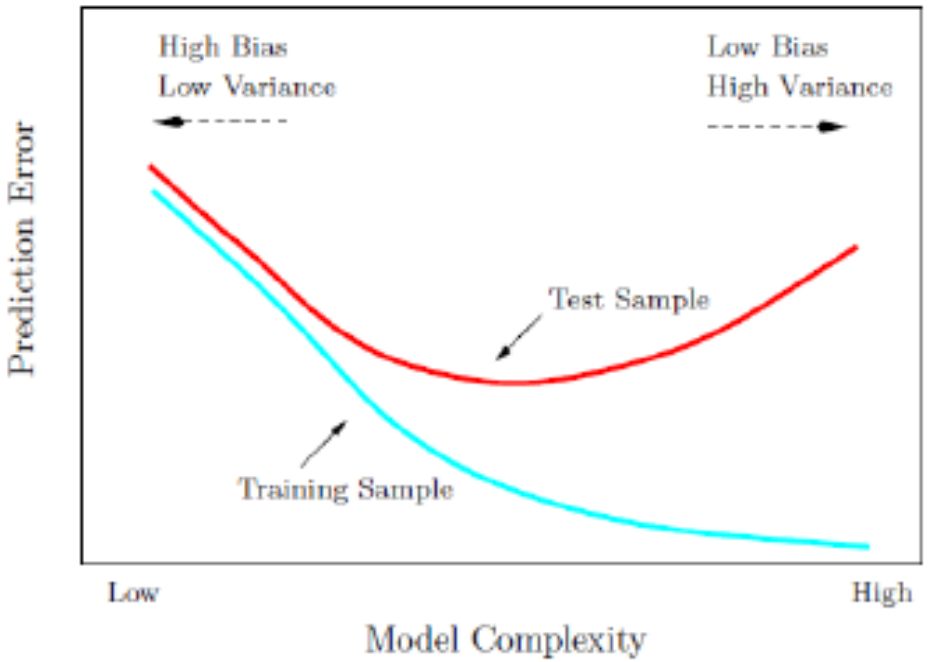
규제화 모델

- 규제화의 필요성

- 변수가 너무 많으면 규제화가 필요함. 일반적으로 샘플 수가 변수 개수보다 적으면 즉 변수의 개수가 샘플 수보다 많으면 회귀모델은 과대적합되는 경향이 있다.

- 과대적합의 의미: 일반화 능력의 결여 (훈련 데이터에서는 예측을 잘 하도록 학습됐으나, 새로운 데이터에 대해서 잘 예측을 못함)





지도 학습에서의 규제화 의미

- 요트 회사의 고객: 단 하나의 변수로 예측하는 경우와 세 개의 변수로 예측하는 경우를 비교해보자.

집이 있는 고객

나이	보유차량수	주택보유	자녀수	혼인상태	애완견	보트구매
66	1	yes	2	사별	no	yes
52	2	yes	3	기혼	no	yes
22	0	no	0	기혼	yes	no
25	1	no	1	미혼	no	no
44	0	no	2	이혼	yes	no
39	1	yes	2	기혼	yes	no
26	1	no	2	미혼	no	no
40	3	yes	1	기혼	yes	no
53	2	yes	2	이혼	no	yes
64	2	yes	3	이혼	no	no
58	2	yes	2	기혼	yes	yes
33	1	no	1	미혼	no	no

45세 이상, 자녀 셋 미만, 이혼하지 않은 고객

나이	보유차량수	주택보유	자녀수	혼인상태	애완견	보트구매
66	1	yes	2	사별	no	yes
52	2	yes	3	기혼	no	yes
22	0	no	0	기혼	yes	no
25	1	no	1	미혼	no	no
44	0	no	2	이혼	yes	no
39	1	yes	2	기혼	yes	no
26	1	no	2	미혼	no	no
40	3	yes	1	기혼	yes	no
53	2	yes	2	이혼	no	yes
64	2	yes	3	이혼	no	no
58	2	yes	2	기혼	yes	yes
33	1	no	1	미혼	no	no

지도 학습

- 데이터셋과 복잡도의 관계

- 많은 변수를 사용하는 것이 더 안정일 수 있다.

- 예: 10,000명의 고객 데이터에서 단지 주택보유 고객이 요트를 구매한다고 하는 모델보다는 45세 이상, 자녀 셋 미만, 이혼하지 않은 고객이 요트를 사려한다면 훨씬 신뢰높은 모델이라고 판단할 수 있다.

릿지 회귀 방정식

강사님은 선형회귀 중 릿지회귀 선호

□ 릿지 (L2 규제화)

- 파라미터 를 선택해 다음 식을 최소화해 회귀계수의 크기를 감소시킨다.

$$\text{mse} + \lambda \sum_{i=1}^{m} b_i^2$$

=

- 람다가 증가하면 어떤 일 일어나는가?

[참고] 싸이킷런에서는 규제화의 강도를 조절하기 위해 alpha를 사용한다.

라쏘 회귀 방정식

□ 라쏘 (L1 규제화)

- 릿지와 유사하나, 다음을 최소화한다.

$$\text{mse} + \lambda \sum_{i=1}^{m} |b_i|$$

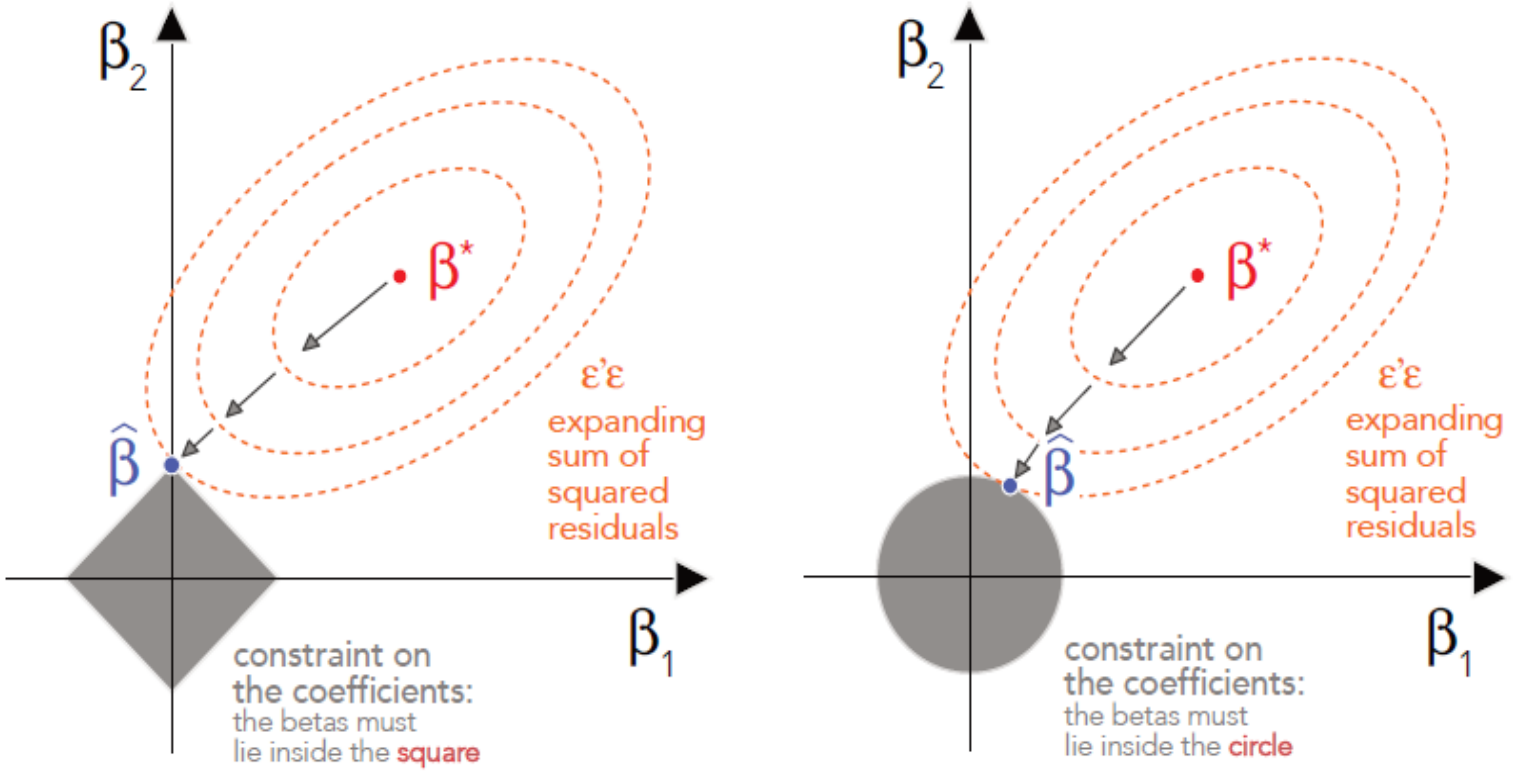
- 덜 중요한 특성을 완전히 제거하는 효과를 갖는다

[참고] 싸이킷런에서는 규제화의 강도를 조절하기 위해 alpha를 사용한다.

라쏘 대 릿지

□ 라쏘 대 릿지

- 많은 특성 중 일부만 중요하다고 판단되면 라쏘를 사용한다. 라쏘의 장점은 해석하기 쉽다.



선형 모델 - 회귀

- ElasticNet

- 릿지와 라쏘의 페널티 결합 (R에서의 glmnet와 같은 것)
- alpha는 규제화의 강도 그리고, l1_ratio 파라미터는 L1 규제와 L2 규제의 양을 조절한다.
- 예를 들어, Lasso는 ElasticNet(l1_ratio=1.0)과 동일하다.

$$MSE + \underbrace{\alpha \times l1\_ratio \sum_{j=1}^{m} |w_j|}_{l_1} + \underbrace{\frac{1}{2}\alpha \times (1 - l1\_ratio) \sum_{j=1}^{m} w_j^2}_{l_1 + l_2}$$

$$\alpha = l_1 + l_2 \qquad l1\_ratio = \frac{l_1}{l_1 + l_2}$$

규제화 모델 구현 예제

- 모델 구현 Sklearn의 Linear_model 패키지에 Ridge()와 Lasso() 모두 있음.
- 모델 인스턴스화

```
from sklearn.Linear_model import Ridge, Lasso
model = Ridge(alpha=1.0)
model = Lasso(alpha=1.0)
```

alpha: 계수 페널티 (λ에 비례한다.)

- 모델 적합화

```
model.fit (X, y)
```

규제화 모델 구현 예제

- 모델 학습과 평가: 보스톤 주택가격 예측

```
from sklearn imort linear_model
Ridge_model = linear_model.Ridge(alpha=1)
Lasso_model = linear_model.Lasso(alpha=1)
Ridge_modelfit(X_train, y_train)
Ridge_modelfit(X_train, y_train)
Lasso_model.fit(X_train, y_train)

#모델 평가
from sklearn_metrics import mean_squared_error
import math
Ridge_predicted = Ridge_model.predict(X_test)
Lasso_predicted=Lasso_model.predict(X_test)
Ridge_RMSE=math.sqrt(mean_squared_error(y_test, Ridge_predicted)
Lasso_RMSE=math.sqrt(mean_squared_error(y_test, Lasso_predicted)
```

[참고] 금융에 이용된 Lasso 기법

The Virtue of Complexity in Return Prediction

BRYAN KELLY, SEMYON MALAMUD, and KANGYING ZHOU*

ABSTRACT

Much of the extant literature predicts market returns with "simple" models that use only a few parameters. Contrary to conventional wisdom, we theoretically prove that simple models severely understate return predictability compared to "complex" models in which the number of parameters exceeds the number of observations. We empirically document the virtue of complexity in U.S. equity market return prediction. Our findings establish the rationale for modeling expected returns through machine learning.

THE FINANCE LITERATURE HAS RECENTLY seen rapid advances in return prediction methods borrowing from the machine learning canon. The primary economic-use case of these predictions has been portfolio construction. While a number of papers document significant empirical gains in portfolio performance through the use of machine learning, there is little theoretical

II. Machine Learning and Random Matrices

The central premise of machine learning is that large data sets can be used in flexible model specifications to improve prediction. This can be understood in the environment above by considering the regime in which the number of predictors, P, is large, perhaps even larger than T. Our main objective is thus to understand the behavior of optimal timing portfolios as the prediction model becomes increasingly complex, that is, as $P \to \infty$. Because this involves estimating infinite-dimensional parameters, traditional large-T asymptotics do not apply, and hence we instead resort to random matrix theory. In this section, we discuss the ridge estimator and present random matrix theory results at the foundation of our theoretical characterization of high-complexity timing strategies.

A. *Least Squares Estimation*

Throughout, we analyze (regularized) least-squares estimators taking the form

$$\hat{\beta}(z) = \left(zI + T^{-1} \sum_t S_t S_t' \right)^{-1} \frac{1}{T} \sum_t S_t R_{t+1}$$

다항식 회귀 모델

다항식 회귀모델

- 다항식 회귀모델

- 선형 모델은 특성과 타겟변수가 선형관계인 경우만 적용되는 것인가?
- 선형 모델의 선형의 의미는 계수와 타겟변수와의 선형관계를 의미하는 것이며, 따라서 특성을 비선형으로 (예: 제곱을 해) 새로운 변수를 만들고 동일한 선형모델을 적용시킬 수 있다. (전처리라고 생각해도 됨)

sklearn 라이브러리 – preprocessing 패키지 – PolynomialFeatures – degree 옵션으로 차원 조절 가능

- 1차원 변수를 2차원 이상의 변수로 확장할 수 있다.

- 예: 2차원의 경우 X_1, X_2, X_3 -> X_1^2, X_2^2, X_3^2, X_1X_2, X_1X_3, X_2X_3

이중 , X_1X_2, X_1X_3, X_2X_3 를 상호작용항(interaction term)이라고 하며, 변수의 상호작용을 다루므로 실무에서 중요한 역할을 하는 경우가 많다.

- 보통 2차원 또는 3차원까지 적용하며, 변수의 수가 많아지므로 Ridge 또는 Lasso를 적용시키는 것이 바람직하다.

다항식 회귀모델 구현예제

- 관련모듈: sklearn.preprocessing

- 모델 인스턴스화

PolynomilaFeature(degree=3, interaction_only=False)

- interaction_only가 True이면 변수간의 곱으로 이뤄진 상호작용만 생성한다.
- 모델 적합화 및 변환
- .fit_transform(X) => 지정된 차수로 확장함.

- 다항식 회귀 모델 구축 예시

```
from sklearn.preprocessing import PolynomilaFeatures
poly=PolynomialFeatures(degree=2)
poly_X_train = poly.fit_transform(X_train)
```

poly_X_test = poly.transform(X_test) (반드시 테스트셋도 훈련세과 동일하게 변화시켜줘야 한다.)