



5장 회귀(Regression)



회귀 소개

- 회귀는 현대 통계학을 이루는 큰 축
- 회귀 분석은 유전적 특성을 연구하던 영국의 통계학자 갈톤(Galton)이 수행한 연구에서 유래했다는 것이 일반론

“부모의 키가 크더라도 자식의 키가 대를 이어 무한정 커지지 않으며, 부모의 키가 작더라도 대를 이어 자식의 키가 무한정 작아지지 않는다”

회귀 분석은 이처럼 데이터 값이 평균과 같은 일정한 값으로 돌아가려는 경향을 이용한 통계학 기법입니다



RSS의 이해

- RSS는 이제 변수가 w_0, w_1 인 식으로 표현할 수 있으며, 이 RSS를 최소로 하는 w_0, w_1 , 즉 회귀 계수를 학습을 통해서 찾는 것이 머신러닝 기반 회귀의 핵심 사항입니다.
- RSS는 회귀식의 독립변수 X, 종속변수 Y가 중심 변수가 아니라 w 변수(회귀 계수)가 중심 변수임을 인지하는 것이 매우 중요합니다(학습 데이터로 입력되는 독립변수와 종속변수는 RSS에서 모두 상수로 간주합니다).
- 일반적으로 RSS는 학습 데이터의 건수로 나누어서 다음과 같이 정규화된 식으로 표현됩니다.

$$RSS(w_0, w_1) = \frac{1}{N} \sum_{i=1}^N (y_i - (w_0 + w_1 * x_i))^2$$

(i는 1부터 학습 데이터의 총 건수 N까지)

RSS의 편미분

$R(w)$ 는 변수가 w 파라미터로 이뤄진 함수이며, $R(w) = \sum_{i=1}^n (y_i - (w_0 + w_1 * x_i))^2$ 입니다.

$R(w)$ 를 미분해 미분 함수의 최솟값을 구해야 하는데, $R(w)$ 는 두 개의 w 파라미터인 w_0 와 w_1 을 각각 가지고 있기 때문에 일반적인 미분을 적용할 수가 없고, w_0 , w_1 각 변수에 편미분을 적용해야 합니다. $R(w)$ 를 최소화하는 w_0 와 w_1 의 값은 각각 $r(w)$ 를 w_0 , w_1 으로 순차적으로 편미분을 수행해 얻을 수 있습니다.

$$\frac{\partial R(w)}{\partial w_1} = \frac{2}{N} \sum_{i=1}^N -x_i * (y_i - (w_0 + w_1 x_i)) = -\frac{2}{N} \sum_{i=1}^N x_i * (\text{실제값}_i - \text{예측값}_i)$$

$$\frac{\partial R(w)}{\partial w_0} = \frac{2}{N} \sum_{i=1}^N -(y_i - (w_0 + w_1 x_i)) = -\frac{2}{N} \sum_{i=1}^N (\text{실제값}_i - \text{예측값}_i)$$

경사 하강법 정리

$$\frac{\partial R(w)}{\partial w_1} = \frac{2}{N} \sum_{i=1}^N -x_i * (y_i - (w_0 + w_1 x_i)) = -\frac{2}{N} \sum_{i=1}^N x_i * (\text{실제값}_i - \text{예측값}_i)$$

$$\frac{\partial R(w)}{\partial w_0} = \frac{2}{N} \sum_{i=1}^N -(y_i - (w_0 + w_1 x_i)) = -\frac{2}{N} \sum_{i=1}^N (\text{실제값}_i - \text{예측값}_i)$$

w1, w0의 편미분 결과값인 $-\frac{2}{N} \sum_{i=1}^N x_i * (\text{실제값}_i - \text{예측값}_i)$ 와 $-\frac{2}{N} \sum_{i=1}^N (\text{실제값}_i - \text{예측값}_i)$ 을 반복적으로 보정하면서 w1, w0 값을 업데이트하면 비용함수 R(W)가 최소가 되는 w1, w0값을 구할 수 있습니다. 하지만 실제로는 위 편미분 값이 너무 클 수 있기 때문에 보정계수 η를 곱하는데, 이를 “학습률”이라고 합니다.

1445185

【개정판】파이썬 머신러닝 완벽 가이드

경사 하강법은 아래와 같은 새로운 w1, 새로운 w0 를 반복적으로 업데이트하면서 비용 함수가 최소가 되는 값을 찾습니다.

$$\text{새로운 } w_1 = \text{이전 } w_1 - (-\eta \frac{2}{N} \sum_{i=1}^N x_i * (\text{실제값}_i - \text{예측값}_i)) = \text{이전 } w_1 + \eta \frac{2}{N} \sum_{i=1}^N x_i * (\text{실제값}_i - \text{예측값}_i)$$

$$\text{새로운 } w_0 = \text{이전 } w_0 - (-\eta \frac{2}{N} \sum_{i=1}^N (\text{실제값}_i - \text{예측값}_i)) = \text{이전 } w_0 + \eta \frac{2}{N} \sum_{i=1}^N (\text{실제값}_i - \text{예측값}_i)$$



사이킷런 LinearRegression 클래스

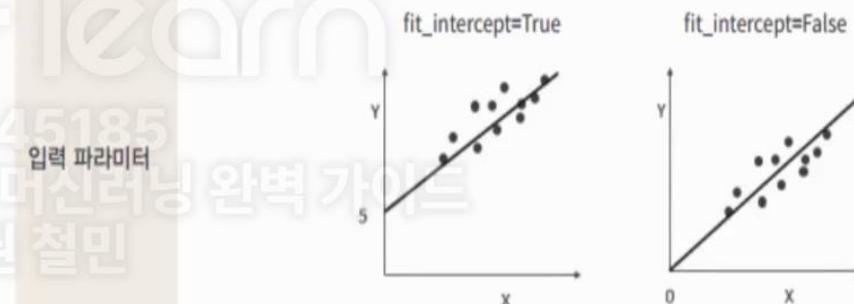
LinearRegression 클래스

```
class sklearn.linear_model.LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=1)
```

LinearRegression 클래스는 예측값과 실제 값의 RSS(Residual Sum of Squares)를 최소화하는 OLS(Ordinary Least Squares) 추정 방식으로 구현한 클래스입니다.

LinearRegression 클래스는 fit() 메서드로 X, y 배열을 입력 받으면 회귀 계수(Coefficients)인 W를 coef_ 속성에 저장합니다.

fit_intercept: 불린 값으로, 디폴트는 True입니다. Intercept(절편) 값을 계산할 것인지 말지를 지정합니다. 만일 False로 지정하면 intercept가 사용되지 않고 0으로 지정됩니다.



normalize: 불린 값으로 디폴트는 False입니다. fit_intercept가 False인 경우에는 이 파라미터가 무시됩니다. 만일 True이면 회귀를 수행하기 전에 입력 데이터 세트를 정규화합니다.

속성

coef_: fit() 메서드를 수행했을 때 회귀 계수가 배열 형태로 저장하는 속성. Shape는 (Target 값 개수, 피처 개수).
intercept_: intercept 값

선형 회귀의 다중 공선성 문제

선형 회귀의 다중 공선성 문제

- 일반적으로 선형 회귀는 입력 피처의 독립성에 많은 영향을 받습니다. 피처간의 상관관계가 매우 높은 경우 분산이 매우 커져서 오류에 매우 민감해집니다. 이러한 현상을 다중 공선성(multi-collinearity) 문제라고 합니다. 일반적으로 상관관계가 높은 피처가 많은 경우 독립적인 중요한 피처만 남기고 제거하거나 규제를 적용합니다.



회귀 평가 지표

평가 지표	설명	수식
MAE	Mean Absolute Error(MAE)이며 실제 값과 예측값의 차이를 절댓값으로 변환해 평균한 것입니다	$MAE = \frac{1}{n} \sum_{i=1}^n Y_i - \hat{Y}_i $
MSE	Mean Squared Error(MSE)이며 실제 값과 예측값의 차이를 제곱해 평균한 것입니다.	$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$
MSLE	MSE에 로그를 적용한 것입니다. 결정값이 클 수록 오류값도 커지기 때문에 일부 큰 오류값들로 인해 전체 오류값이 커지는 것을 막아줍니다.	$\begin{aligned} MSLE \\ = \frac{1}{n} \sum_{i=1}^n (\log(Y_i + 1) - \log(\hat{Y}_i + 1))^2 \end{aligned}$
RMSE	MSE 같은 오류의 제곱을 구하므로 실제 오류 평균보다 더 커지는 특성이 있으므로 MSE에 루트를 씌운 것이 RMSE(Root Mean Squared Error)입니다	$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2}$
RMSLE	RMSE에 로그를 적용한 것입니다. 결정값이 클 수록 오류값도 커지기 때문에 일부 큰 오류값들로 인해 전체 오류값이 커지는 것을 막아줍니다.	$\begin{aligned} RMSLE \\ = \sqrt{\frac{1}{n} \sum_{i=1}^n (\log(Y_i + 1) - \log(\hat{Y}_i + 1))^2} \end{aligned}$
R²	분산 기반으로 예측 성능을 평가합니다. 실제 값의 분산 대비 예측값의 분산 비율을 지표로 하며, 1에 가까울수록 예측 정확도가 높습니다.	$R^2 = \frac{\text{예측값 Variance}}{\text{실제 값 Variance}}$

MAE와 RMSE의 비교

MAE에 비해 RMSE는 큰 오류값에 상대적인 페널티를 더 부여 합니다.

예를 들어 다섯개의 오류값(실제값과 예측값의 차이)이 10, 20, 10, 10, 100과 같이 다른 값에 비해 큰 오류값이 존재하는 경우 RMSE는 전반적으로 MAE보다 높습니다.

$$\text{MAE} = (10 + 20 + 10 + 10 + 100)/5 = 30 \text{ 이고 RMSE는 } \sqrt{(100 + 400 + 100 + 100 + 10000)/5} = \sqrt{2140} = 46.26$$

이처럼 상대적으로 더 큰 오류값이 있을 때 이에 대한 페널티를 더 부과하는 평가 방식이 RMSE입니다.



사이킷런 회귀 평가 API

- 다음은 각 평가 방법에 대한 사이킷런의 API 및 cross_val_score나 GridSearchCV에서 평가 시 사용되는 scoring 파라미터의 적용 값입니다

평가 방법	사이킷런 평가 지표 API	Scoring 함수 적용 값
MAE	metrics.mean_absolute_error	'neg_mean_absolute_error'
MSE	metrics.mean_squared_error	'neg_mean_squared_error'
RMSE	metrics.mean_squared_error를 그대로 사용하되 squared 파라미터를 False로 설정.	'neg_root_mean_squared_error'
MSLE	metrics.mean_squared_log_error	'neg_mean_squared_log_error'
R^2	metrics.r2_score	'r2'

과거 버전의 사이킷런은 RMSE를 계산하는 함수가 제공되지 않았지만, 0.22 버전부터는 RMSE를 위한 함수를 제공합니다. RMSE를 구하기 위해서는 MSE를 위한 metrics.mean_squared_error() 함수를 그대로 사용하되, squared 파라미터를 False로 지정하여 사용합니다. mean_squared_error() 함수는 squared 파라미터가 기본적으로 True입니다. 즉 MSE는 사이킷런에서 mean_squared_error(실제값, 예측값, squared=True)이며 RMSE는 mean_squared_error(실제값, 예측값, squared=False)를 이용해서 구합니다.

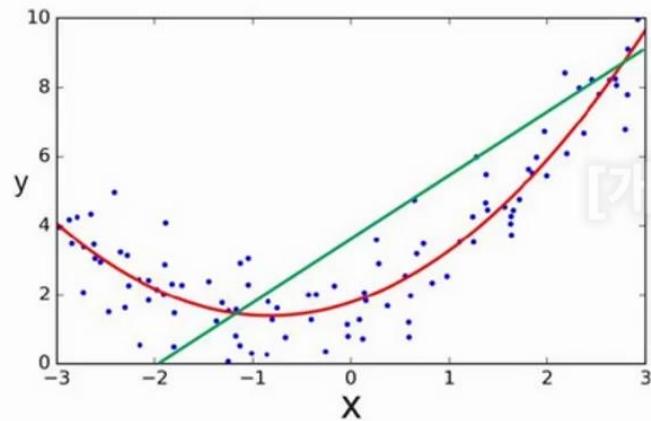
사이킷런 Scoring 함수에 회귀 평가 적용 시 유의 사항

cross_val_score, GridSearchCV와 같은 Scoring 함수에 회귀 평가 지표를 적용 시 유의 사항

- MAE의 사이킷런 scoring 파라미터 값은 'neg_mean_absolute_error' 입니다. 이는 Negative(음수) 값을 가진다는 의미인데, MAE는 절댓값의 합이기 때문에 음수가 될 수 없습니다.
- Scoring 함수에 'neg_mean_absolute_error'를 적용해 음수값을 반환하는 이유는 사이킷런의 Scoring 함수가 score값이 클수록 좋은 평가 결과로 자동 평가하기 때문입니다. 따라서 -1을 원래의 평가 지표 값에 곱해서 음수(Negative)를 만들어 작은 오류 값이 더 큰 숫자로 인식하게 합니다. 예를 들어 $10 > 1$ 이지만 음수를 곱하면 $-1 > -10$ 이 됩니다.
- metrics.mean_absolute_error()와 같은 사이킷런 평가 지표 API는 정상적으로 양수의 값을 반환합니다. 하지만 Scoring 함수의 scoring 파라미터 값 'neg_mean_absolute_error'가 의미하는 것은 $-1 * \text{metrics.mean_absolute_error}()$ 이니 주의가 필요합니다.

다항회귀 개요(Polynomial Regression)

다항 회귀는 $y = w_0 + w_1*x_1 + w_2*x_2 + w_3*x_1*x_2 + w_4*x_1^2 + w_5*x_2^2$ 과 같이 회귀식이 독립변수의 단항식이 아닌 2차, 3차 방정식과 같은 다항식으로 표현되는 것을 지칭합니다.



데이터 세트에 대해서 피처 X에 대해 Target Y 값의 관계를 단순 선형 회귀
직선형으로 표현한 것보다 다항 회귀 곡선형으로 표현한 것이 더 예측 성능이
높습니다.

선형 회귀와 비선형 회귀의 구분

선형 회귀

$$Y = w_0 + w_1 * x_1 + w_2 * x_2 + w_3 * x_1 * x_2 + w_4 * x_1^2 + w_5 * x_2^2$$

새로운 변수인 Z 를 $Z = [x_1, x_2, x_1 * x_2, x_1^2, x_2^2]$ 로 한다면

$$Y = w_0 + w_1 * z_1 + w_2 * z_2 + w_3 * z_3 + w_4 * z_4 + w_5 * z_5$$

비선형 회귀

$$Y = w_1 * \cos(X + w_4) + w_2 * \cos(2*X + w_4) + w_3$$

$$Y = w_1 * X^{w_2}$$

다항 회귀는 선형 회귀입니다. 회귀에서 선형 회귀/비선형 회귀를 나누는 기준은 회귀 계수가 선형/비선형인지에 따른 것이지 독립변수의 선형/비선형 여부와는 무관합니다

사이킷런에서의 다항 회귀

사이킷런은 다항회귀를 바로 API로 제공하지 않습니다.

대신 `PolynomialFeatures` 클래스로 원본 단항 피처들을 다항 피처들로 변환한 데이터 세트에 `LinearRegression` 객체를 적용하여 다항회귀 기능을 제공합니다.

PolynomialFeatures

원본 피처 데이터 세트를 기반으로 `degree` 차수에 따른 다항식을 적용하여 새로운 피처들을 생성하는 클래스
피처 엔지니어링의 기법중의 하나임.

단항 피처 $[x_1, x_2]$ 를 $\text{Degree} = 2$, 즉 2차 다항 피처로 변환한다면?

$(x_1 + x_2)^2$ 의 식 전개에 대응되는 $[1, x_1, x_2, x_1x_2, x_1^2, x_2^2]$ 의

다항 피처들로 변환

단항 피처 $[x_1, x_2]$ 를 $\text{Degree} = 3$, 즉 3차 다항 피처로 변환한다면?

$(x_1 + x_2)^3$ 의 식 전개에 대응되는

$[1, x_1, x_2, x_1x_2, x_1^2, x_2^2, x_1^3, x_2^3, x_1^2x_2, x_1x_2^2, x_2^3]$ 의 다항 피처들로 변환

1차 단항 피처들의 값이 $[x_1, x_2] = [0 1]$ 일 경우

2차 다항 피처들의 값은 $[1, x_1 = 0, x_2 = 1, x_1x_2 = 0, x_1^2 = 0, x_2^2 = 1]$

형태인 $[1, 0, 1, 0, 0, 1]$ 로 변환



사이킷런에서의 다항 회귀

사이킷런은 다항회귀를 바로 API로 제공하지 않습니다.

대신 PolynomialFeatures 클래스로 원본 단항 피처들을 다항 피처들로 변환한 데이터 세트에 LinearRegression 객체를 적용하여 다항회귀 기능을 제공합니다.



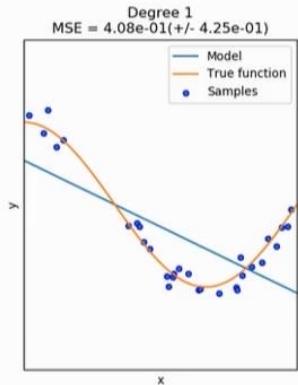
단항 피처 $[x_1, x_2]$ 를 2차 다항 피처 $[1, x_1, x_2, x_1x_2, x_1^2, x_2^2]$ 로 변경
(degree=2로 가정)

PolynomialFeatures로 변환된 x 피처들을 LinearRegression
객체로 학습

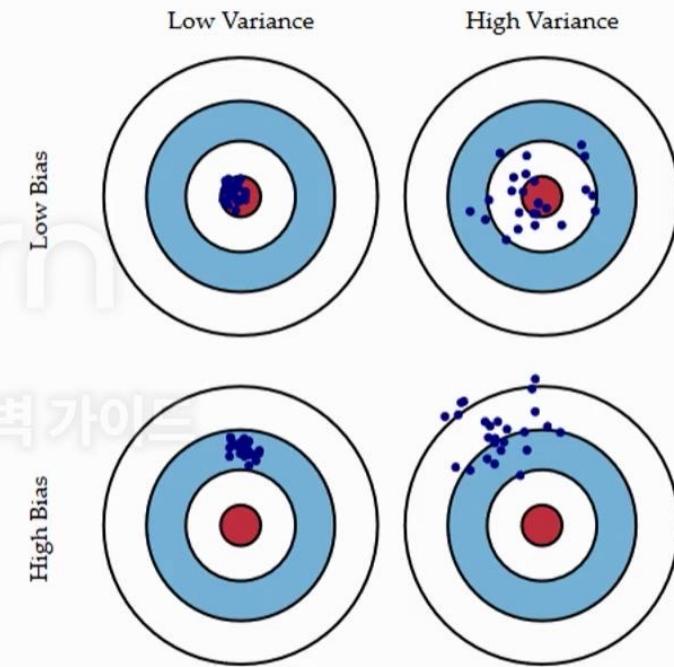
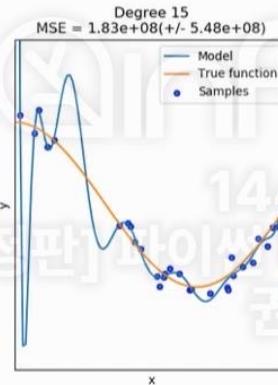
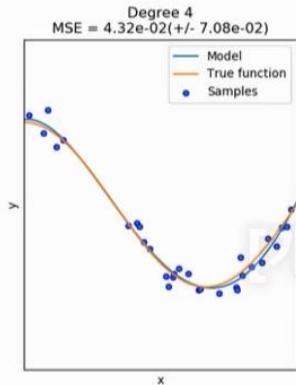
사이킷런에서는 일반적으로 Pipeline 클래스를 이용하여
PolynomialFeatures 변환과 LinearRegression 학습/예측을 결합하여 다항 회귀를 구현합니다.

편향-분산 트레이드 오프(Bias-Variance Trade off)

과소적합



과대 적합

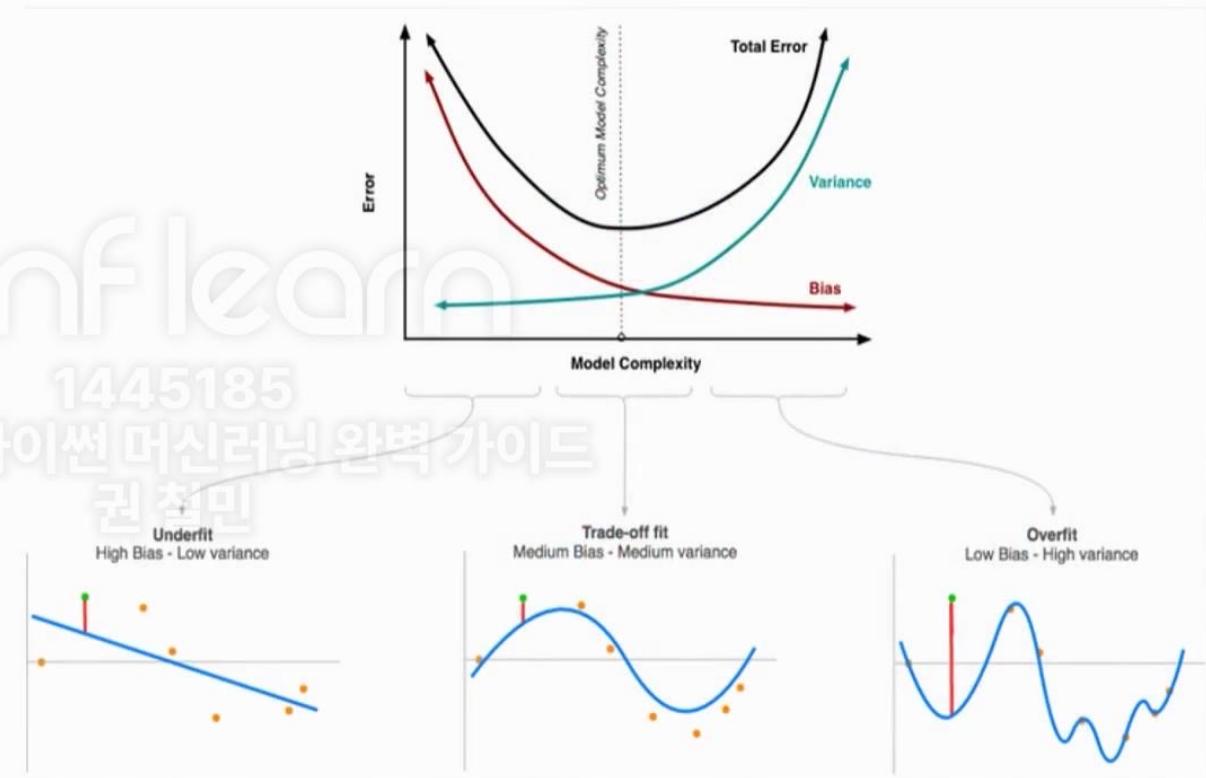


< 편향과 분산의 고/저에 따른 표현. <http://scott.fortmann-roe.com/docs/BiasVariance.html>에서 발췌 >

편향-분산 트레이드 오프(Bias-Variance Trade off)

편향이 높으면 분산은 낮아지고
↑ ↓
과소적합

분산이 높으면 편향이 낮아짐
↑ ↓
과대적합



< 편향과 분산에 따른 전체 오류 값(Total Error) 곡선.

<http://scott.fortmann-roe.com/docs/BiasVariance.html>에서 발췌. >

규제 선형 회귀 개요

앞의 예제에서 Degree=15의 다항회귀는 지나치게 모든 데이터에 적합한 회귀식을 만들기 위해서 다항식이 복잡해지고 회귀 계수가 매우 크게 설정이 되면서 과대적합이 되고 평가 데이터 세트에 대해서 형편없는 예측 성능을 보였습니다. 따라서 회귀 모델은 적절히 데이터에 적합하면서도 회귀 계수가 기하급수적으로 커지는 것을 제어할 수 있어야 합니다.



규제 선형 모델에서 alpha의 역할

- alpha가 0(또는 매우 작은 값)이라면 비용 함수 식은 기준과 동일한 $\text{Min}(\text{RSS}(W) + 0)$ 이 될 것입니다.
- 반면에 alpha가 무한대(또는 매우 큰 값)라면 비용 함수 식은 $\text{RSS}(W)$ 에 비해 $\alpha * ||W||_2^2$ 값이 너무 커지게 되므로 W 값을 0(또는 매우 작게)으로 만들어야 Cost가 최소화되는 비용 함수 목표를 달성할 수 있습니다.
- 즉, alpha 값을 크게 하면 비용 함수는 회귀 계수 W의 값을 작게 해 과적합을 개선할 수 있으며 alpha 값을 작게 하면 회귀 계수 W의 값이 커져도 어느 정도 상쇄가 가능하므로 학습 데이터 적합을 더 개선할 수 있습니다

비용 함수 목표 = $\text{Min} (\text{RSS}(W) + \alpha * ||W||_2^2)$



- $\alpha = 0$ 인 경우는 W가 커도 $\alpha * ||W||_2^2$ 가 0이 되어 비용 함수는 $\text{Min}(\text{RSS}(W))$
- α = 무한대인 경우 $\alpha * ||W||_2^2$ 도 무한대가 되므로 비용 함수는 W를 0에 가깝게 최소화 해야 함.

릿지(Ridge) 회귀

- 릿지 회귀는 alpha값을 이용하여 회귀 계수의 크기를 조절합니다(alpha값이 크면 회귀 계수 값이 작아지고, alpha값이 작으면 회귀 계수 값이 커집니다)
- 사이킷런은 릿지 회귀를 위해 Ridge 클래스를 제공합니다.

라쏘(Lasso) 회귀

W 의 절댓값에 페널티를 부여하는 L1 규제를 선형 회귀에 적용한 것이 라쏘(Lasso) 회귀입니다. 즉 L1 규제는 $\alpha * \|W\|_1$ 을 의미하며, 라쏘 회귀 비용함수의 목표는 $RSS(W) + \alpha * \|W\|_1$ 식을 최소화 하는 W 를 찾는 것입니다. L2규제가 회귀 계수의 크기를 감소시키는 데 반해, **L1규제는 불필요한 회귀 계수를 급격하게 감소시켜 0으로 만들고 제거합니다.** 이러한 측면에서 L1 규제는 적절한 피처만 회귀에 포함시키는 피처 셀렉션의 특성을 가지고 있습니다.

사이킷런은 Lasso 클래스를 통해 라쏘 회귀를 구현하였습니다

엘라스틱넷(Elastic Net) 회귀

- 엘라스틱넷(Elastic Net) 회귀는 L2 규제와 L1 규제를 결합한 회귀입니다. 따라서 엘라스틱넷 회귀 비용함수의 목표는 $RSS(W) + \alpha_2 * \|W\|_2^2 + \alpha_1 * \|W\|_1$ 식을 최소화 하는 W 를 찾는 것입니다.
- 엘라스틱넷은 라쏘 회귀가 서로 상관관계가 높은 피처들의 경우에 이들 중에서 중요 피처만을 셀렉션하고 다른 피처들은 모두 회귀 계수를 0으로 만드는 성향이 강합니다. 특히 이러한 성향으로 인해 α 값에 따라 회귀 계수의 값이 급격히 변동 할 수도 있는데, 엘라스틱넷 회귀는 이를 완화하기 위해 L2 규제를 라쏘 회귀에 추가한 것입니다.

[개정판] 파이썬 머신러닝 완벽 가이드
권철민

사이킷런 엘라스틱넷 회귀

사이킷런은 ElasticNet 클래스를 통해서 엘라스틱넷 회귀를 구현합니다.

ElasticNet 클래스의 주요 생성 파라미터는 alpha 와 l1_ratio 입니다. ElasticNet 클래스의 alpha는 Ridge와 Lasso 클래스의 alpha값과는 다릅니다.

엘라스틱넷의 규제는 $a * L1\text{규제} + b * L2\text{규제}$ 로 정의될 수 있습니다.

ElasticNet alpha 파라미터

ElasticNet l1_ratio 파라미터

이 때 a는 L1규제의 alpha값, b는 L2 규제의 alpha 값입니다.

따라서 ElasticNet 클래스의 alpha파라미터 값은 $a + b$ 입니다

ElasticNet 클래스의 l1_ratio 파라미터 값은 $a / (a + b)$ 입니다.

l1_ratio가 0 이면 a 가 0 이므로 L2 규제와 동일합니다.

l1_ratio가 1이면 b가 0 이므로 L1 규제와 동일합니다.

$0 < l1_ratio < 1$ 이며 L1과 L2 규제를 함께 적절히 적용합니다.

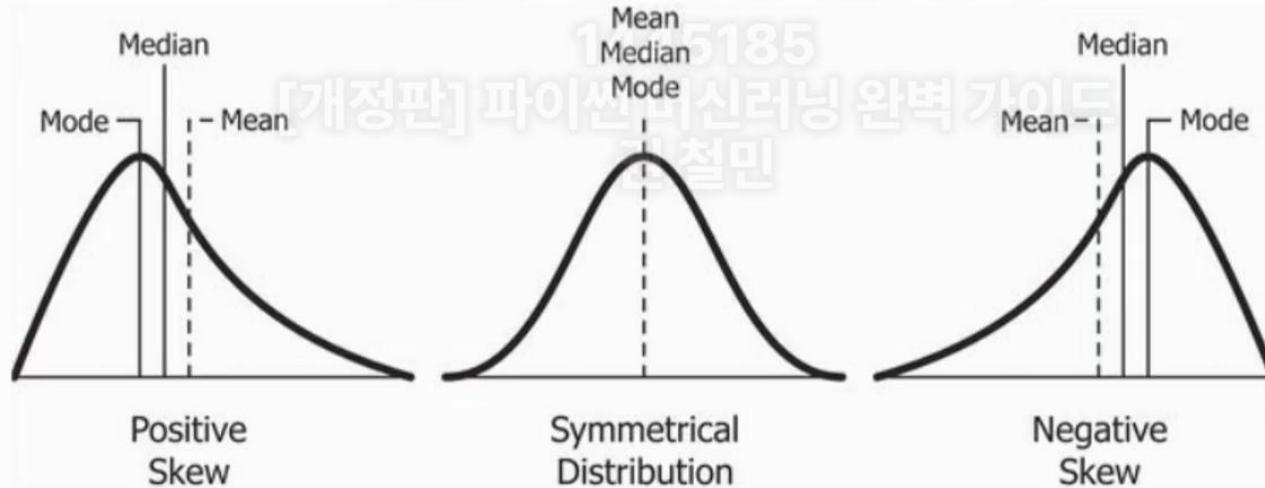
만일 ElasticNet의 alpha가 10, l1_ratio가 0.7 이라면

$L1_ratio = 0.7 = a / a + b = 7/10$ 이므로 $a=7$ 이고 L1 alpha값은 7, L2 alpha값은 3입니다.



선형 회귀 모델을 위한 데이터 변환

- 선형 회귀 모델은 일반적으로 피처와 타겟값 간에 선형의 관계가 있다고 가정하고 이러한 최적의 선형 함수를 찾아내 결과 값을 예측합니다.
- 선형 회귀 모델은 피처값과 타겟값의 분포가 정규 분포 형태를 선호합니다.



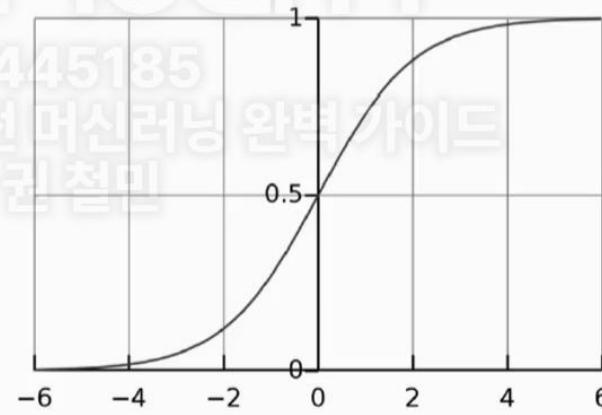
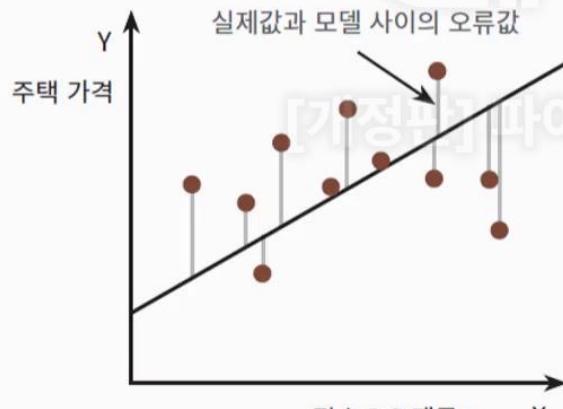
선형 회귀를 위한 데이터 변환 방법

변환 대상	설명
타겟값 변환	타겟값은 정규 분포 선호. Skew되어 있을 경우 주로 로그 변환을 적용
피처값 변환 – 스케일링(Scaling)	피처들에 대한 균일한 스케일링/정규화 적용. StandardScaler를 이용하여 표준 정규 분포 형태 변환 또는 MinMaxScaler를 이용하여 최소값 0, 최대값 1로 변환
피처값 변환 – 다항 특성 변환	스케일링/정규화 수행한 데이터 세트에 다시 다항 특성(Polynomial Feature)을 적용하여 변환
피처값 변환 – 로그 변환	왜도(Skewness)가 심한 중요 피처들에 대해서 로그 변환을 적용. 일반적으로 많이 사용됨.

로지스틱 회귀(Logistic Regression) 개요

로지스틱 회귀는 선형 회귀 방식을 분류에 적용한 알고리즘입니다. 즉, 로지스틱 회귀는 분류에 사용됩니다.

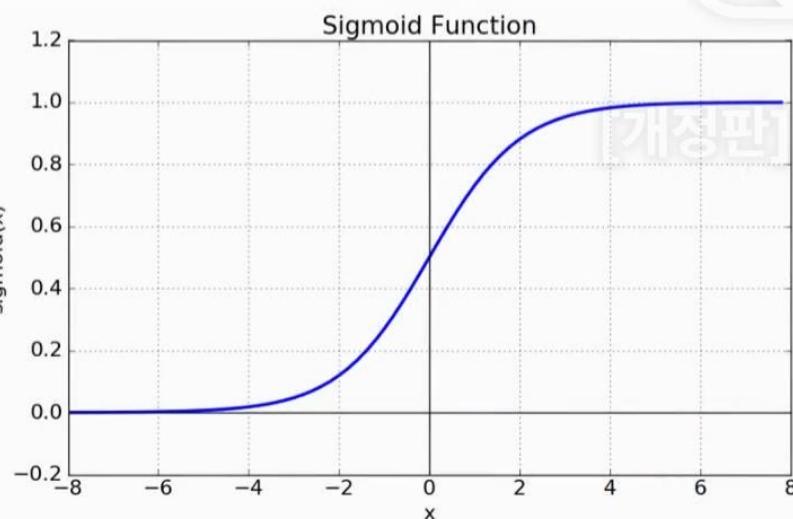
로지스틱 회귀가 선형 회귀와 다른 점은 선형 함수의 회귀 최적선을 찾은 것이 아니라, 시그모이드(Sigmoid) 함수의 최적선을 찾고 이 시그모이드 함수의 반환 값을 확률로 간주해 확률에 따라 분류를 결정한다는 것입니다.



로지스틱 회귀 예측

로지스틱 회귀는 주로 이진 분류(0과 1)에 사용됩니다(물론 다중 클래스 분류에도 적용 될 수 있습니다). 로지스틱 회귀에서 예측 값은 예측 확률을 의미하며, 예측 값 즉 예측 확률이 0.5 이상이면 1로, 0.5 이하이면 0으로 예측합니다. 로지스틱 회귀의 예측 확률은 시그모이드 함수의 출력값으로 계산됩니다.

시그모이드 함수 $f(x) = \frac{1}{1 + e^{-x}}$



단순 선형 회귀: $y = w_1x + w_0$ 가 있다고 할 때

로지스틱 회귀는 0과 1을 예측하기에 단순 회귀식에 적용할 수는 없습니다. 하지만 Odds(성공확률/실패확률)를 통해 선형 회귀식에 확률을 적용합니다.

$$\text{Odds}(p) = p/(1-p)$$

하지만 확률 p 의 범위가 0 ~ 1 사이이고, 선형 회귀의 반환값인 무한대 ~ +무한대 값에 대응하기 위해서 로그 변환을 수행하고 아래와 같이 선형회귀를 적용합니다.

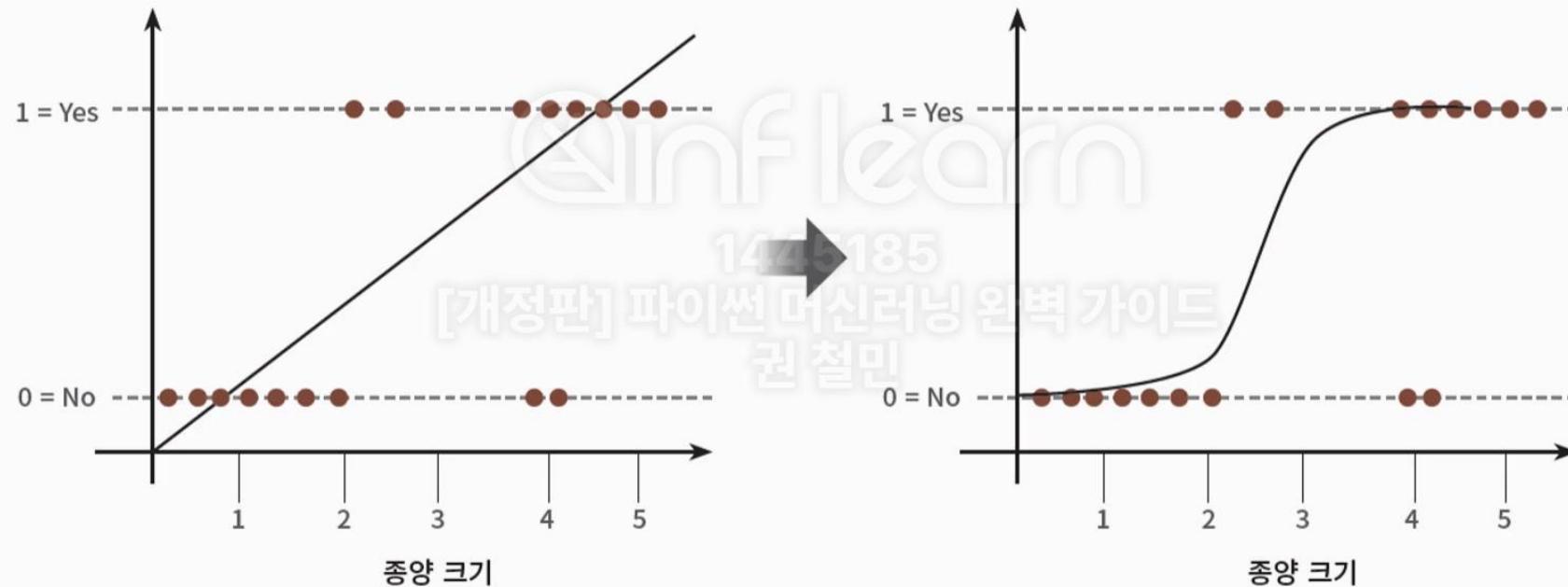
$$\text{Log(Odds}(p)) = w_1x + w_0$$

해당 식을 데이터 값 x 의 확률 p 로 정리하면 아래와 같습니다.

$$p(x) = \frac{1}{1 + e^{-(w_1x + w_0)}}$$

로지스틱 회귀는 학습을 통해 시그모이드 함수의 w 를 최적화하여 예측하는 것입니다

시그모이드를 이용한 로지스틱 회귀 예측



로지스틱 회귀 특징

- 로지스틱 회귀는 가볍고, 빠르며, 이진 분류 예측 성능도 뛰어납니다. 특히 희소한 데이터 세트 분류에서 성능이 좋아서 텍스트 분류에 자주 사용됩니다.

1445185
[개정판] 파이썬 머신러닝 완벽 가이드
권철민



사이킷런 로지스틱 회귀

- 사이킷런은 로지스틱 회귀를 LogisticRegression 클래스로 구현합니다.
- LogisticRegression의 주요 하이퍼 파라미터로 penalty, C, solver가 있습니다. Penalty는 규제 유형을 설정하며 'l2'로 설정 시 L2 규제를, 'l1'으로 설정 시 L1 규제를 뜻합니다. C는 규제 강도를 조절하는 alpha값의 역수입니다. 즉 $C = 1/\alpha$ 입니다. C값이 작을 수록 규제 강도가 큽니다.
- solver는 회귀 계수 최적화를 위한 다양한 최적화 방식입니다.



LogisticRegression의 solver 유형

- lbfgs: 사이킷런 버전 0.22부터 solver의 기본 설정값입니다. 메모리 공간을 절약할 수 있고, CPU 코어 수가 많다면 최적화를 병렬로 수행할 수 있습니다.
- liblinear: 사이킷런 버전 0.21까지에서 solver의 기본 설정값입니다. 다차원이고 작은 데이터 세트에서 효과적으로 동작하지만 국소 최적화(Local Minimum)에 이슈가 있고, 병렬로 최적화할 수 없습니다.
- newton-cg : 좀 더 정교한 최적화를 가능하게 하지만, 대용량의 데이터에서 속도가 많이 느려집니다.
- sag: Stochastic Average Gradient로서 경사 하강법 기반의 최적화를 적용합니다. 대용량의 데이터에서 빠르게 최적화합니다.
- saga: sag와 유사한 최적화 방식이며 L1 정규화를 가능하게 해줍니다.



회귀 트리 개요

- 사이킷런의 결정 트리 및 결정 트리 기반의 앙상블 알고리즘은 분류 뿐만 아니라 회귀도 가능합니다.
- 이는 트리가 CART(Classification and Regression Tree)를 기반으로 만들어 졌기 때문입니다. CART는 분류 뿐만 아니라 회귀도 가능한 트리 분할 알고리즘입니다.
- CART 회귀 트리는 분류와 유사하게 분할을 하며, 최종 분할이 완료된 후에 각 분할 영역에 있는 데이터 결정값들의 평균 값으로 학습/예측 합니다.

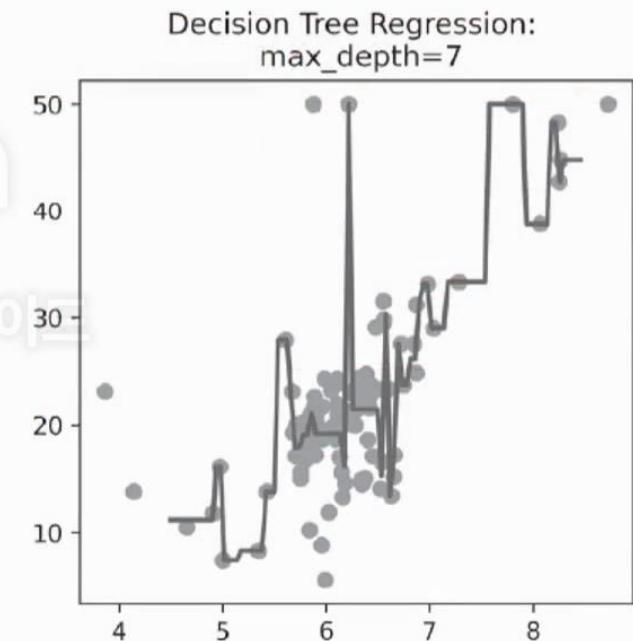
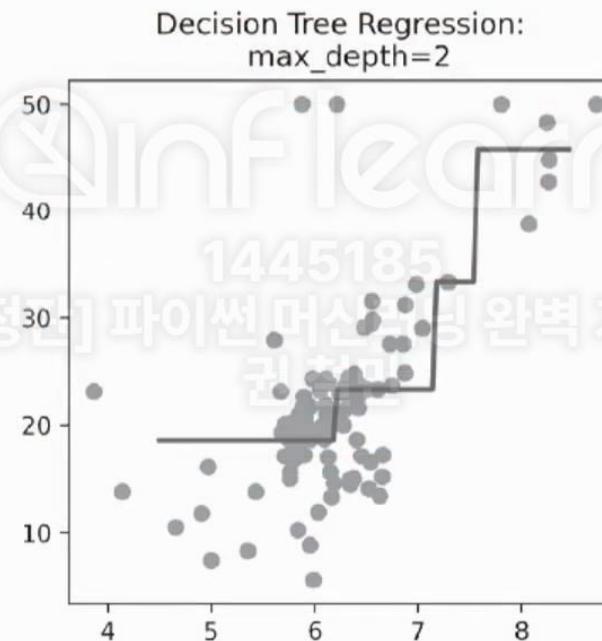
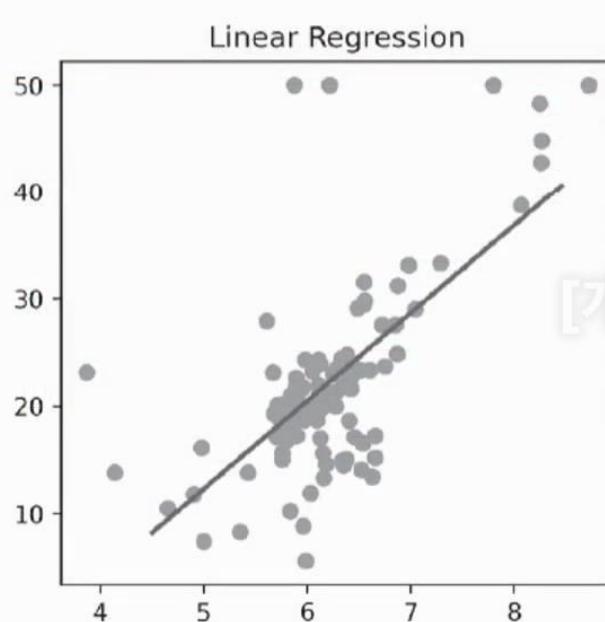
사이킷런의 회귀 트리 지원

알고리즘	회귀 Estimator 클래스	분류 Estimator 클래스
Decision Tree	DecisionTreeRegressor	DecisionTreeClassifier
Gradient Boosting	GradientBoostingRegressor	GradientBoostingClassifier
XGBoost	XGBRegressor	XGBClassifier
LightGBM	LGBMRegressor	LGBMClassifier



회귀 트리의 오버 피팅

회귀 트리 역시 복잡한 트리 구조를 가질 경우 오버 피팅하기 쉬우므로 트리의 크기와 노드 개수의 제한 등의 방법을 통해 오버 피팅을 개선 할 수 있습니다.



회귀 Summary

- 선형 회귀와 비용 함수 RSS
- 경사 하강법
- 다항회귀와 과소적합/과대적합
- 규제 – L2규제를 적용한 릿지, L1규제를 적용한 라쏘, L1과 L2규제가 결합된 엘라스틱넷 회귀
- 분류를 위한 로지스틱 회귀
- CART기반의 회귀 트리
- 왜곡도 개선을 위한 데이터 변환과 원-핫 인코딩
- 실습 예제를 통한 데이터 정제와 변환 그리고 선형 회귀/회귀 트리/혼합모델/스태킹 모델 학습/예측/평가 비교