

머신러닝 강의자료

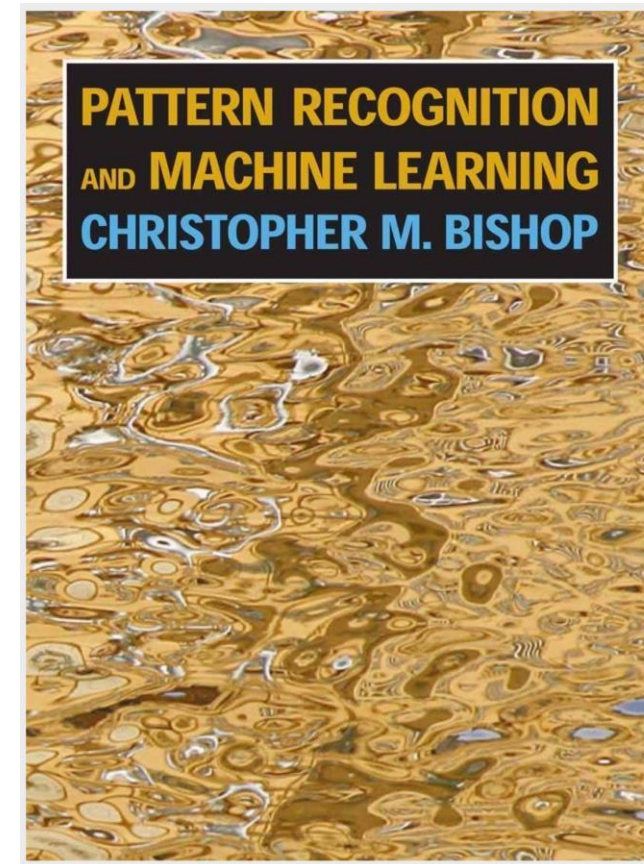
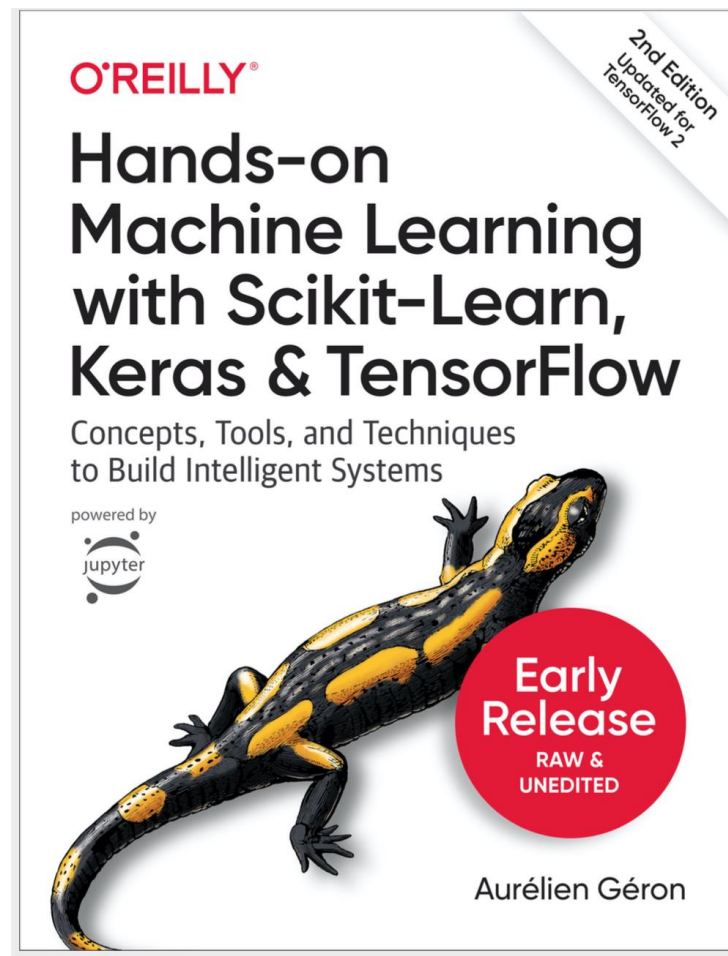
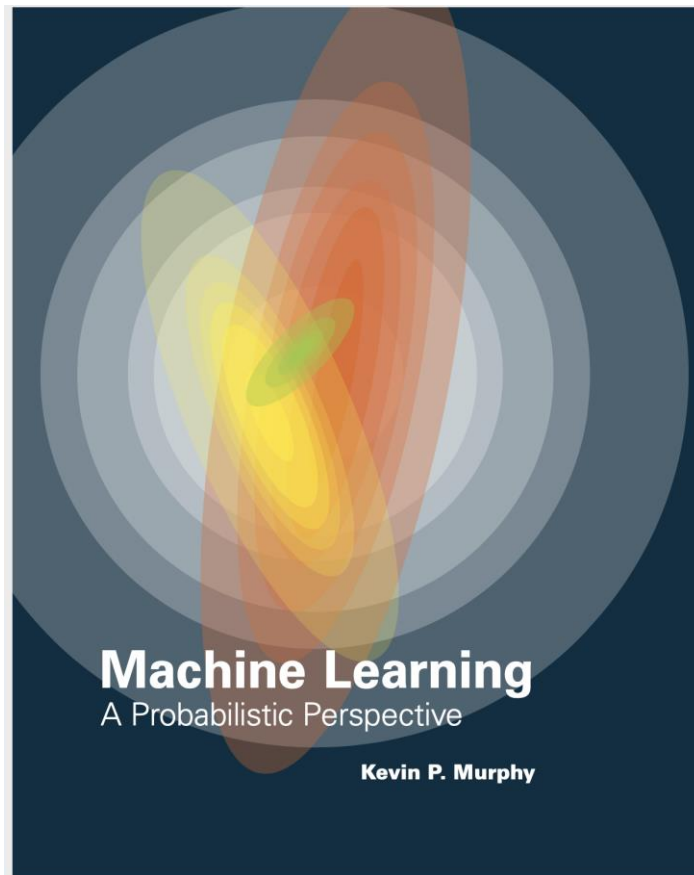
4강 - 회귀분석

닥터윌컨설팅 딥러닝 R&D 책임연구원
고려대학교 인공지능대학원 박사과정

류회성(Hoe Sung Ryu)



들어가기 앞서



들어가기 앞서

● GitHub

- <https://github.com/hoesungryu/blockchain-devML-course>

hoesungryu / blockchain-devML-course

Watch 1 Unstar 1 Fork 1

<> Code Issues Pull requests Actions Projects Wiki Security Insights

master 1 branch 0 tags Go to file Add file Code

hoesungryu and hoesungryu 2일차 업데이트 ec864ee 2 days ago 9 commits

| File/Folder | Commit Message | Time Ago |
|------------------|----------------|------------|
| code | 2일차 업데이트 | 2 days ago |
| data | 1일차 자료 업로드 | 3 days ago |
| img | 2일차 업데이트 | 2 days ago |
| lecture_note | 2일차 업데이트 | 2 days ago |
| .DS_Store | 2일차 업데이트 | 2 days ago |
| .gitignore | Initial commit | 4 days ago |
| LICENSE | Initial commit | 4 days ago |
| README.md | 1일차 자료 업로드 | 3 days ago |
| requirements.txt | 2일차 업데이트 | 2 days ago |

README.md

블록체인 초안여게 고우귀저

About
No description, website, or topics provided.

Readme
MIT License

Releases
No releases published
[Create a new release](#)

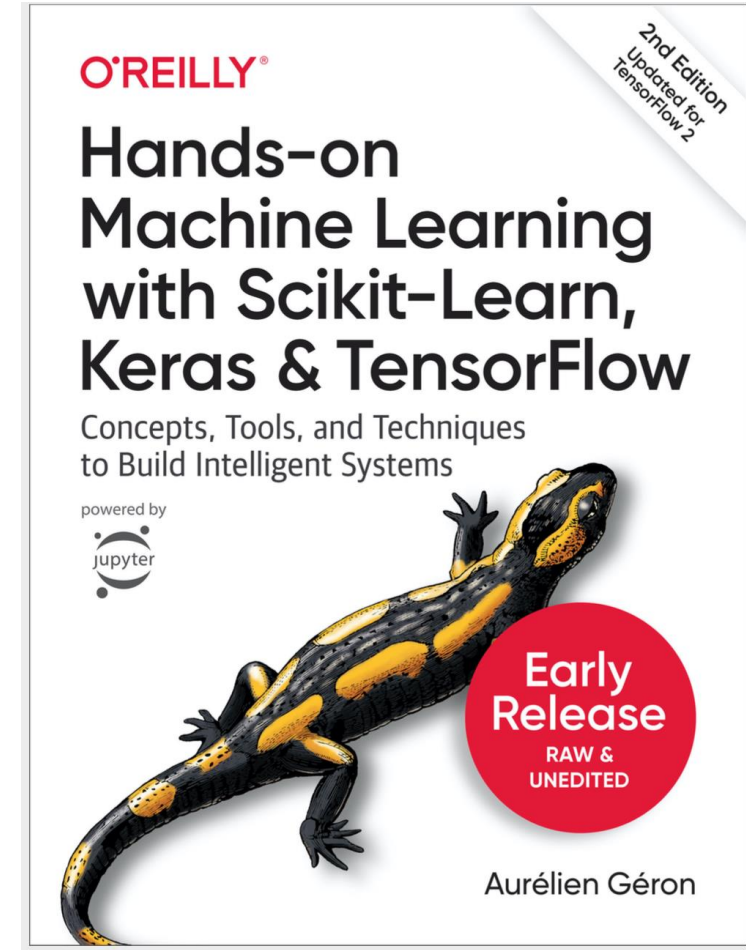
Packages
No packages published
[Publish your first package](#)

Contributors 2
mssung94 mssung94

강의내용

● 오늘 배울 내용

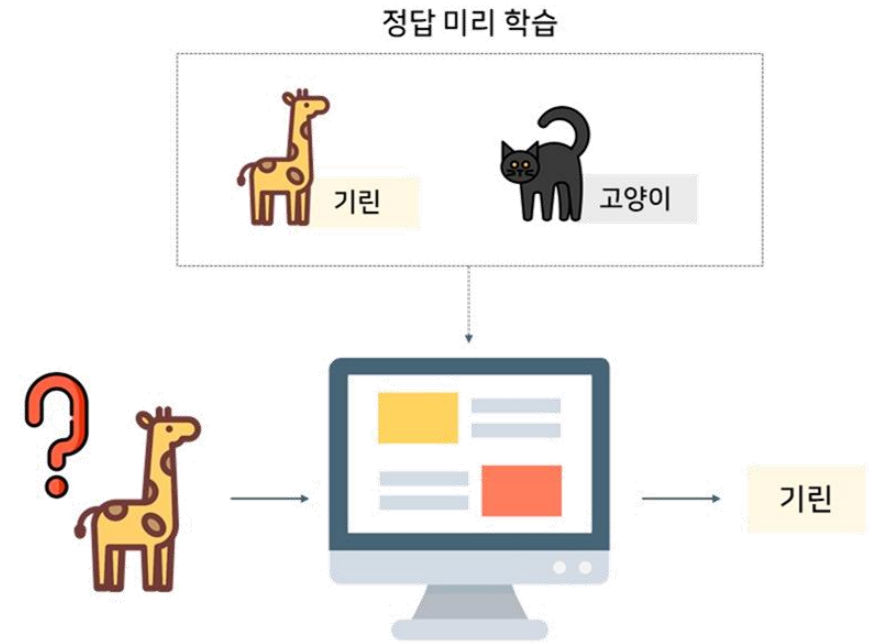
- 선형 회귀
- 경사 하강법
- 다항 회귀
- 학습 곡선
- 규제가 있는 선형 모델
- 로지스틱 회귀




[지난 시간 복습]

지도 학습

- 훈련 데이터에 **정답이 있음**
- 전형적인 지도 학습 작업
 - 분류(Classification)
 - 회귀(Regression)
- 대표적인 지도 학습 알고리즘
 - k-최근접 이웃(k-Nearest Neighbors)
 - 선형 회귀(Linear Regression)
 - 로지스틱 회귀(Logistic Regression)
 - 서포트 벡터 머신(Support Vector Machine)
 - 결정 트리(Decision Tree)과 랜덤 포레스트(Random Forest)
 - 신경망(Neural Network)



머신러닝 파이프라인


[Install](#)
[User Guide](#)
[API](#)
[Examples](#)
[More ▾](#)

scikit-learn

Machine Learning in Python

[Getting Started](#)
[Release Highlights for 0.23](#)
[GitHub](#)

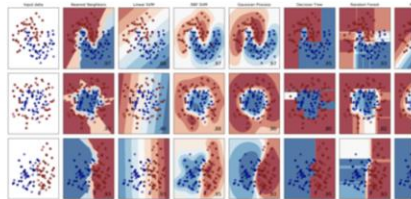
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



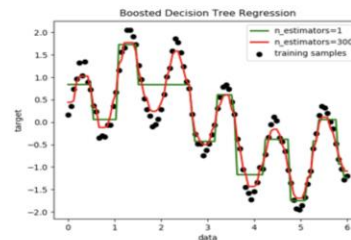
Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...



Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.

머신러닝 파이프라인

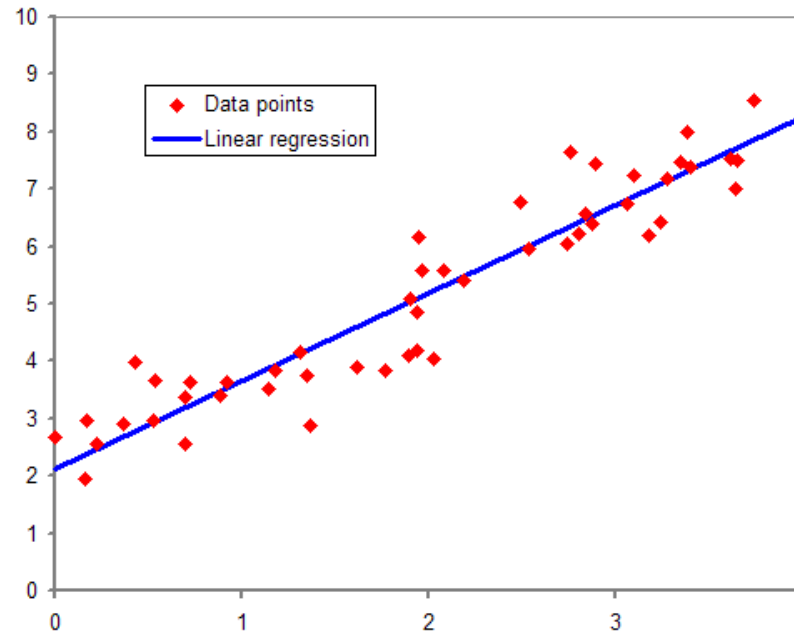
| 분류 | 모듈명 | 설명 |
|----------------------|---|---|
| 예제 데이터 | <code>sklearn.datasets</code> | Scikit-learn에 내장된 예제 데이터셋 |
| 피쳐 처리 | <code>sklearn.preprocessing</code> | 데이터 전처리에 필요한 다양한 기능 제공(인코딩, 정규화, 스케일링 등) |
| | <code>sklearn.feature_selection</code> | 알고리즘에 영향을 미치는 특성을 우선순위로 선택 작업을 수행하는 다양한 기능 제공 |
| | <code>sklearn.feature_extraction</code> | 텍스트 데이터나 이미지 데이터의 벡터화된 피쳐를 추출하는데 사용됨. |
| | <code>sklearn.decomposition</code> | 차원 축소와 관련된 알고리즘 지원 (PCA, NMF 등) |
| 데이터 분리, 검증 & 파라미터 튜닝 | <code>sklearn.model_selection</code> | 교차 검증을 위한 훈련용/테스트용 데이터 분리, GridSearch 기능 등 제공 |
| 평가 | <code>sklearn.metrics</code> | 각종 머신 러닝 알고리즘(회귀, 분류, 클러스터링 등)의 다양한 성능 측정 방법 제공 (accuracy, precision, recall, ROC-AUC 곡선 등) |
| 머신 러닝 알고리즘 | <code>sklearn.ensemble</code> | 앙상블 알고리즘 제공(Random Forest, AdaBoost, Gradient boosting, 등) |
| | <code>sklearn.linear_model</code> | 선형회귀, 릿지, 라쏘 및 로지스틱 회귀 등 회귀 관련 알고리즘 지원 |
| | <code>sklearn.naïve_bayes</code> | 나이브 베이즈 알고리즘 제공. 가우시안 NB, 다항 분포 NB 등 |
| | <code>sklearn.neighbors</code> | KNN 알고리즘 제공 |
| | <code>sklearn.svm</code> | SVM 알고리즘 제공 |
| | <code>sklearn.tree</code> | 의사 결정 트리 알고리즘 제공 |
| | <code>sklearn.cluster</code> | 클러스터링 알고리즘 제공 (k-means, DBScan 등) |
| 유틸리티 (지원 기능) | <code>sklearn.pipeline</code> | 피쳐 처리 등의 변환과 ML 알고리즘 학습, 예측 등을 함께 묶어서 실행할 수 있는 유틸리티 제공 |



[선형 회귀]

선형 회귀

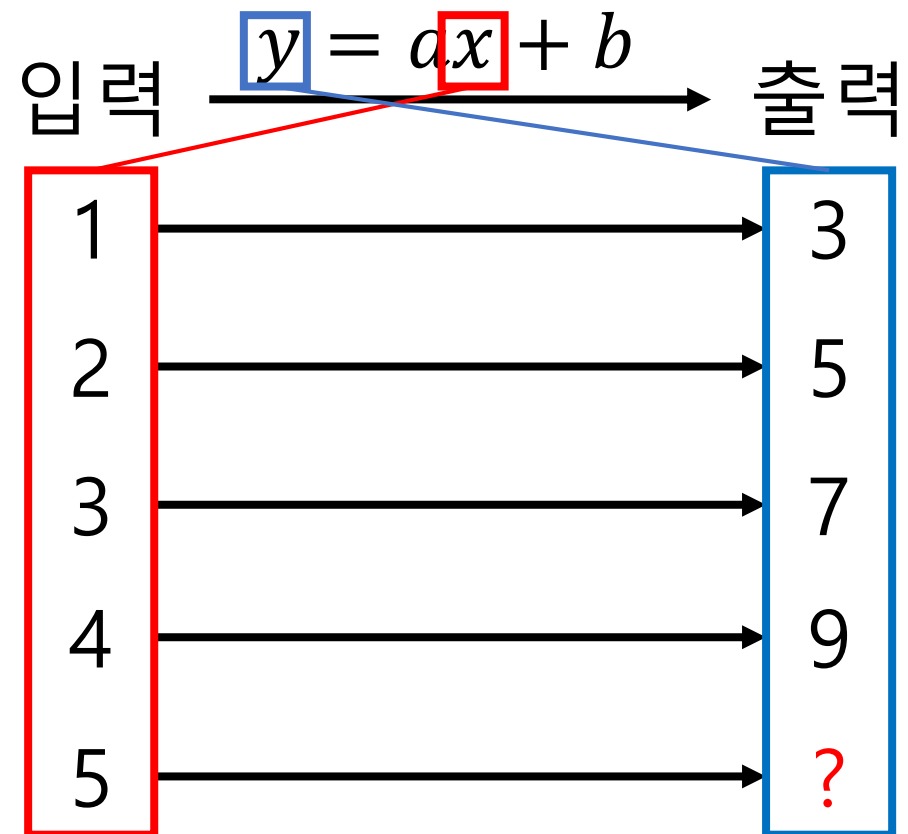
- 종속 변수 y 와 한 개 이상의 독립변수 x 와의 상관관계를 모델링 하는 분석기법
- 변수들 간의 상관관계를 파악하여, 어떤 특정 변수의 값을 다른 변수값을 이용하여 설명/ 예측하는 기법



선형 회귀

| 입력 | | 출력 |
|----|---|----|
| 1 | → | 3 |
| 2 | → | 5 |
| 3 | → | 7 |
| 4 | → | 9 |
| 5 | → | ? |

선형 회귀

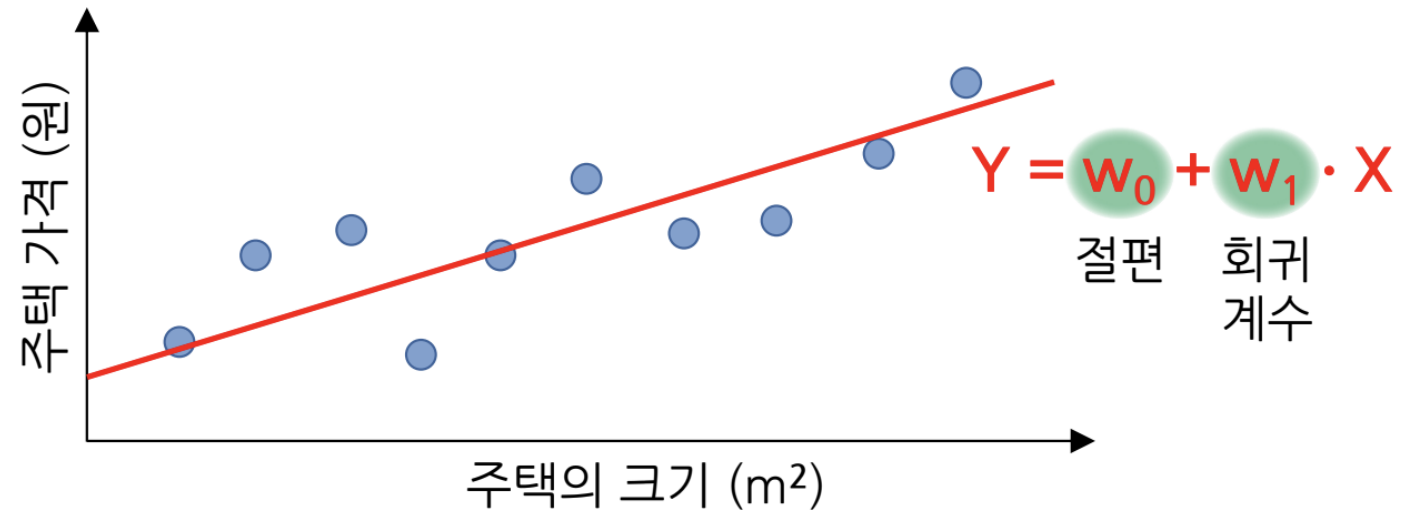


선형 회귀

| 입력 | $y = 2x + 1$ | 출력 |
|----|--------------|----|
| 1 | → | 3 |
| 2 | → | 5 |
| 3 | → | 7 |
| 4 | → | 9 |
| 5 | → | 11 |

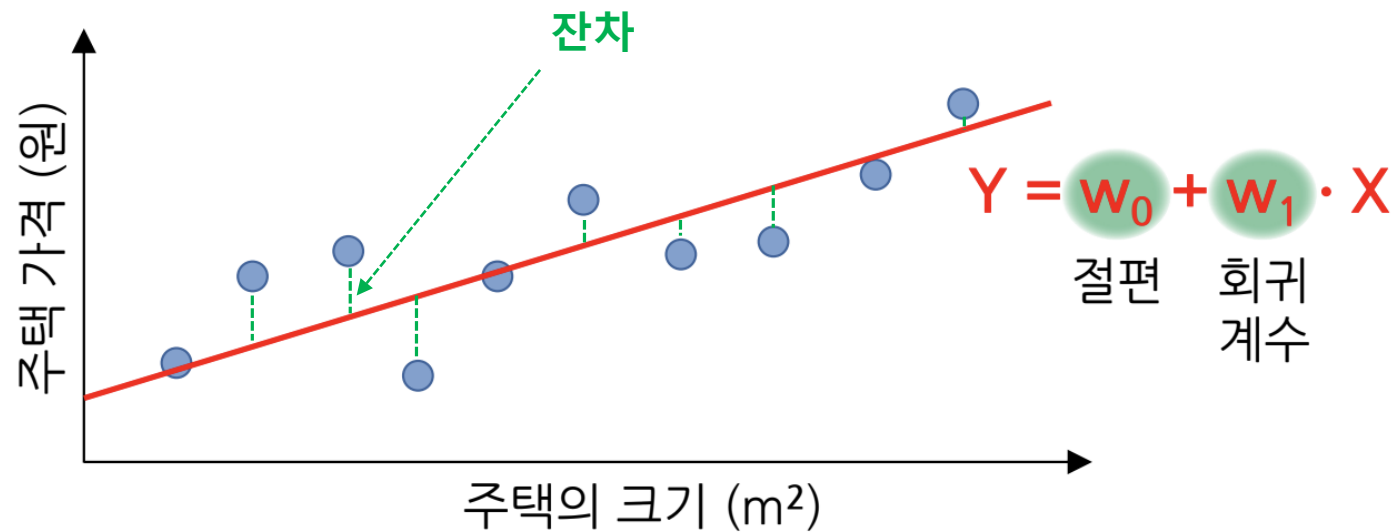
단순 선형 회귀

- 독립변수 X 와 종속변수 Y 의 관계를 $Y = w_0 + w_1X$ 형태의 1차 함수식으로 표현 가능
- 회귀계수(Coefficient)
 - 독립변수가 종속변수에 끼치는 영향력의 정도로서 직선의 기울기를 뜻한다.
- 절편(intercept)
 - 독립변수가 0일 때의 상수 값



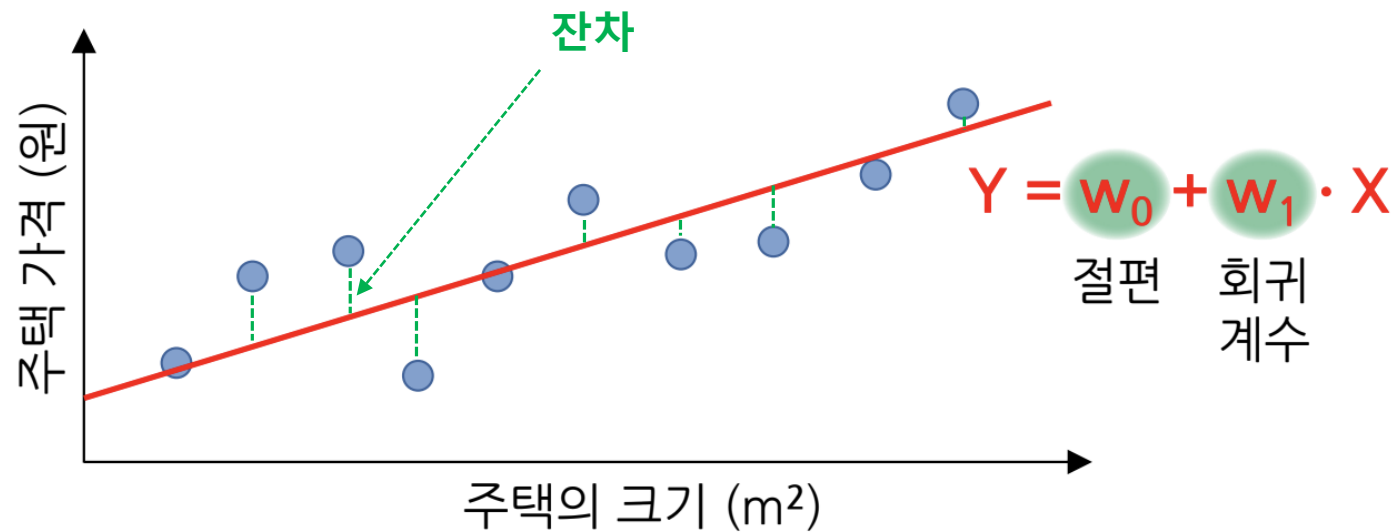
단순 선형 회귀

- 독립변수 X 와 종속변수 Y 의 관계($Y = w_0 + w_1 X$)를 어떻게 찾을 수 있을까?
 - 잔차(Residual)의 총 합이 가장 작은 1차 함수식을 찾자



단순 선형 회귀

- 잔차(Residual)의 총합을 회귀분석에서 손실함수(Loss function)라 한다.
- 최적의 회귀 모형을 만든다는 것은 이러한 손실함수가 최소가 되는 회귀계수를 구한다는 의미이다.



단순 선형 회귀

- 회귀 분석에 대한 주요 평가 지표는 다음과 같다.

| 평가 지표 | 의미 | 수식 | 모듈 |
|------------------------------|--------------------------------|---------------------------------------|--|
| MAE (Mean Absolute Error) | 실제값과 예측값 차이의 절대값들의 평균 | $\sum y - \hat{y} $ | sklearn.metric 에서 mean_absolute_error |
| MSE (Mean Squared Error) | 실제값과 예측값 차이의 제곱의 평균 | $\sum (y - \hat{y})^2$ | sklearn.metric 에서 mean_squared_error |
| RMSE (Root MSE) | MSE의 제곱근 값 | \sqrt{MSE} | np.sqrt(MSE) |
| R^2 | 결정 계수 실제값의 분산대비 예측값의 분산의 비율 | $\frac{\text{예측값 분산}}{\text{실제값 분산}}$ | sklearn.metric 에서 r2_score |

단순 선형 회귀

● 결정계수(Coefficient of Determination) 란?

- 회귀식이 얼마나 정확한지 나타내는 지표
 - 변수 X, Y에 대하여 각각의 크기(단위)에 영향을 받지 않도록 단위를 보정한 값

$$R^2 = \frac{\text{예측값 분산}}{\text{실제값 분산}} = \frac{\sum(\hat{y}-m)^2}{\sum(m-y)^2} \quad (\text{이 때, } \hat{y} \text{ 는 예측값, } y \text{ 는 실제값, } m \text{ 평균})$$

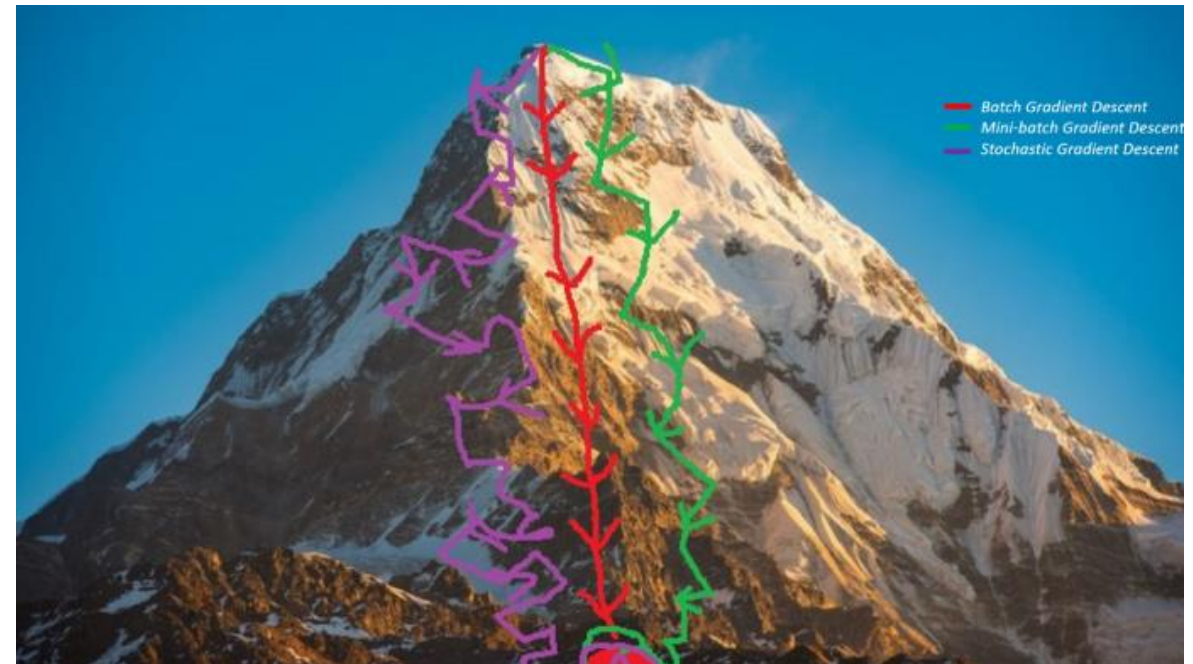
- 결정 계수의 값은 $0 \leq R^2 \leq 1$ 이며,
1 에 가까울수록 설명력이 강하고, 0 에 가까울수록 설명력이 약하다.
일반적으로 결정계수의 값이 0.65 이상 이면 설명력이 있다고 판단한다.

[실습]

[경사 하강법]

경사하강법

- 주어진 문제의 비용함수(Cost function)의 결과 값을 최소화 하도록 반복하여 매개변수를 조정해 가는 최적화(optimization) 기법
 - 늦은 밤 산 속에서 길을 잃었다면, 매 위치에서 가장 가파른 경사를 따라서 아래로 내려가는 것이 좋다.



경사 하강법

● 기본 아이디어

- 비용 함수를 최소화하기 위해 여러 번 반복해서 파라미터를 조정해 나가자.
- 변수 θ 에 대하여 경사(gradient)를 계산하여 비용 함수의 값이 감소되는 방향으로 진행

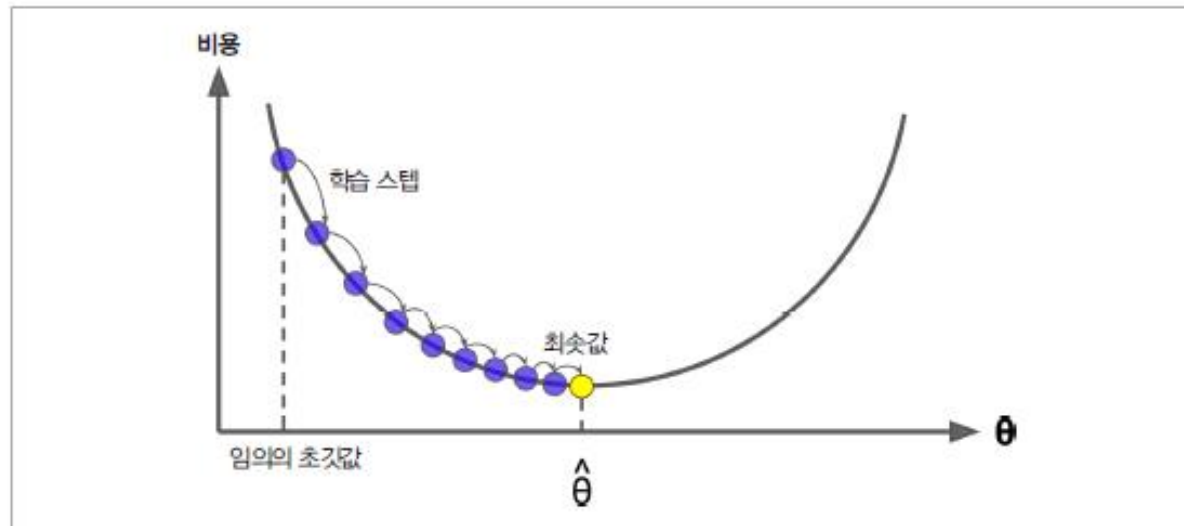


그림 4-3 이 경사 하강법 그림에서 모델 파라미터가 무작위하게 초기화된 후 반복적으로 수정되어 비용 함수를 최소화 합니다. 학습 스텝 크기는 비용 함수의 기울기에 비례합니다. 따라서 파라미터가 최솟값에 가까워질수록 스텝 크기가 점진적으로 줄어듭니다.

경사 하강법

● 학습률

- 스텝 사이즈
- 얼마만큼의 크기로 학습시킬건지

● 학습률이 너무 작을 때

- 수렴하기 위해서 반복을 많이 진행해야 하므로 시간이 오래 걸림

● 학습률이 너무 클 때

- 수렴하지 못하고 더 큰 값으로 발산하게 되므로 학습이 되지 않음

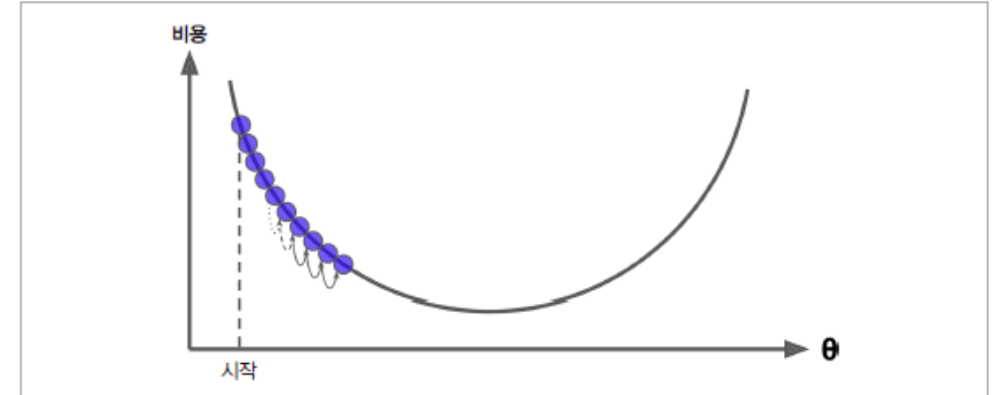


그림 4-4 학습률이 너무 작을 때

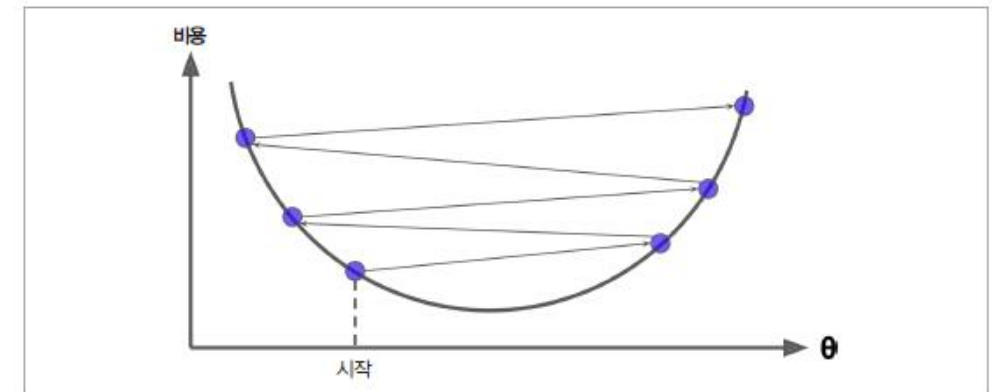


그림 4-5 학습률이 너무 클 때

경사 하강법

● 경사 하강법의 문제점

- 무작위 초기화 때문에 전역 최솟값이 아닌 지역 최솟값으로 수렴
- 평탄한 지역을 지나기 위해 시간이 오래 걸리고 일찍 멈추게 됨

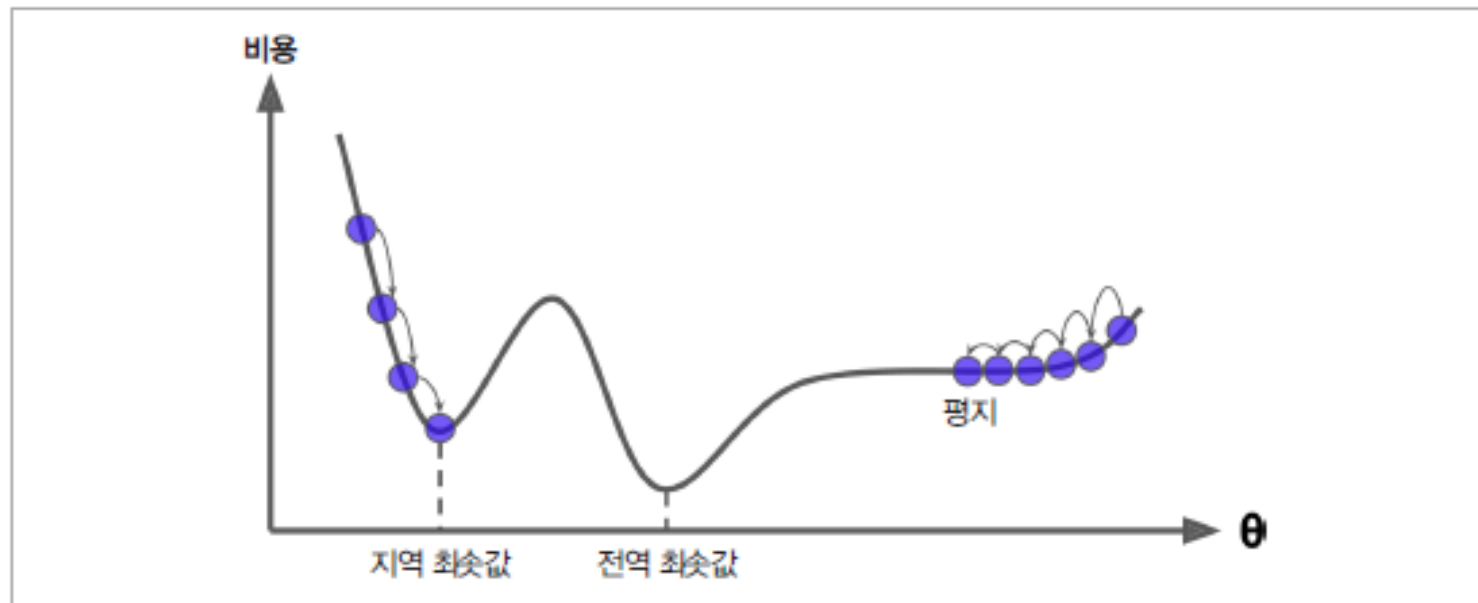


그림 4-6 경사 하강법의 문제점

경사 하강법

● 3가지 종류의 경사 하강법

- 배치 경사 하강법
 - 매 경사 하강법 스텝에서 전체 훈련 세트에 대해서 그래디언트를 계산
 - 전체 훈련 세트를 사용하기 때문에 느림
- 확률적 경사 하강법
 - 매 반복마다 딱 하나의 샘플만으로 그래디언트를 계산하여 업데이트함
 - 학습이 매우 불안정하게 됨
- 미니배치 경사 하강법
 - 위 두가지 방법의 단점을 보완
 - 적당한 사이즈만큼 샘플을 가져와서 그래디언트를 업데이트

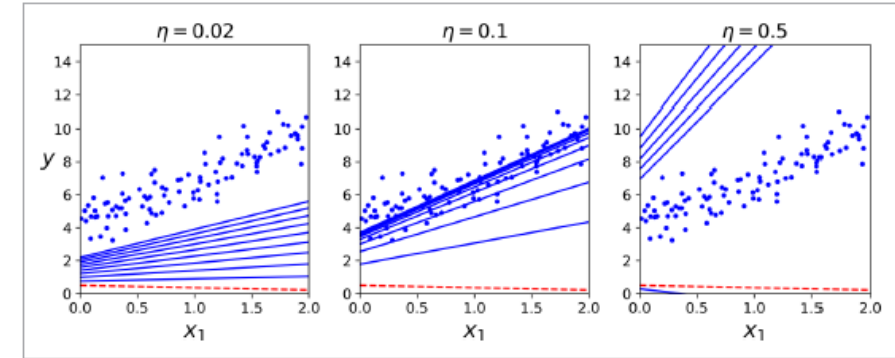


그림 4-8 여러 가지 학습률에 대한 경사 하강법

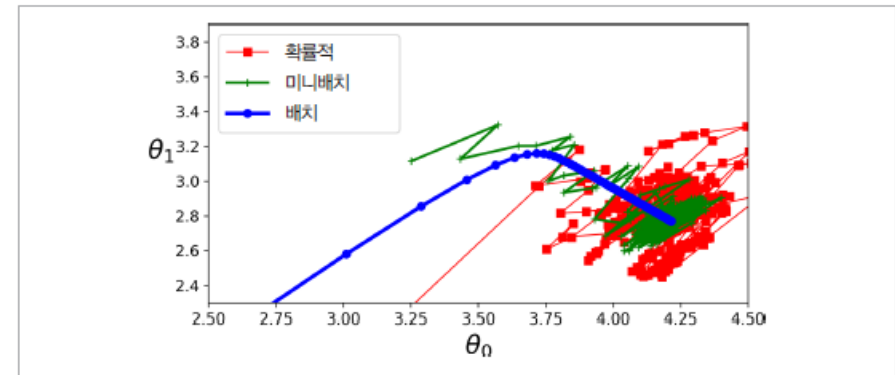


그림 4-11 파라미터 공간에 표시된 경사 하강법의 경로

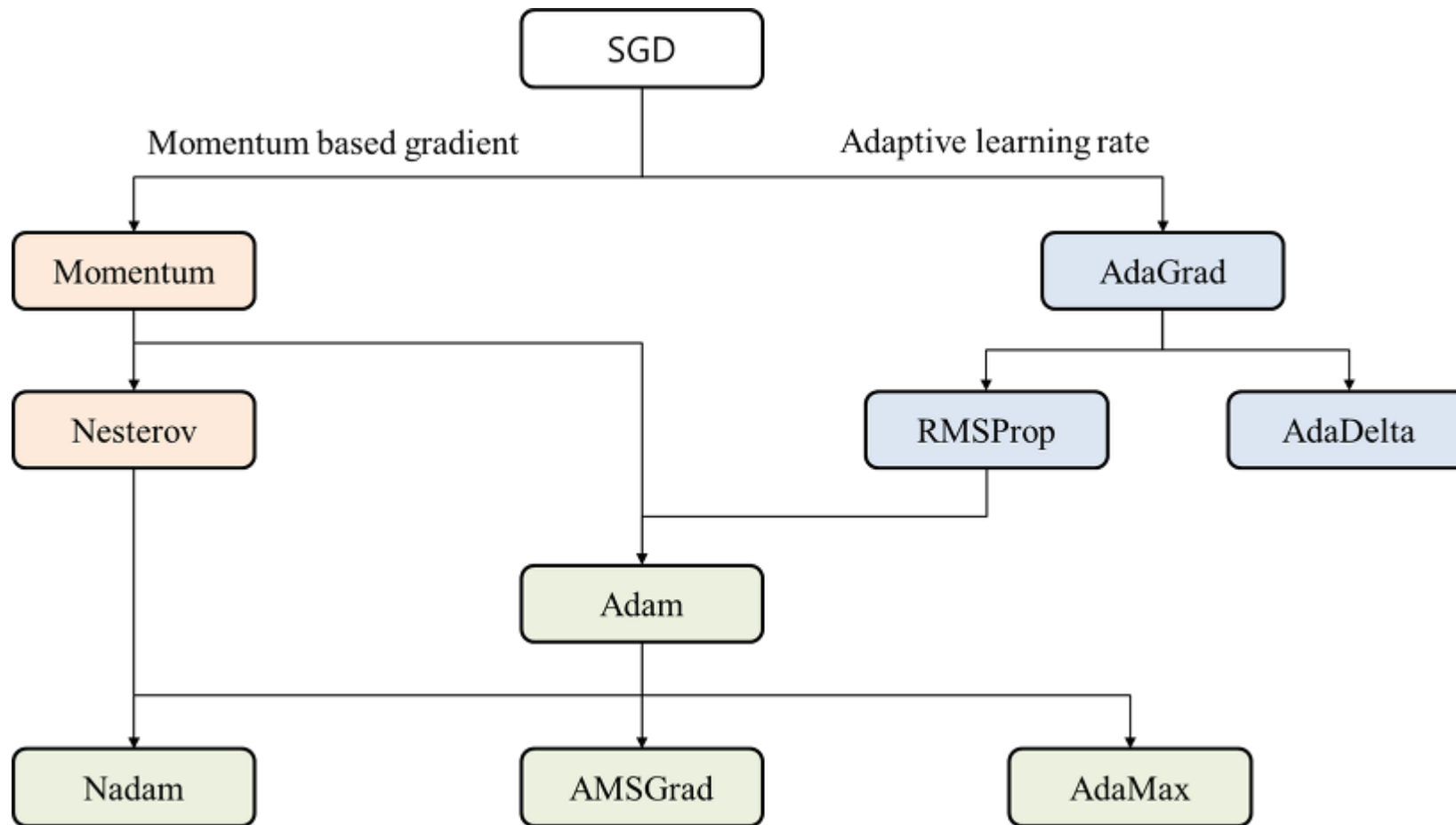
경사 하강법

● 선형 회귀를 사용했을 때 알고리즘 비교

표 4-1 선형 회귀를 사용한 알고리즘 비교²⁰

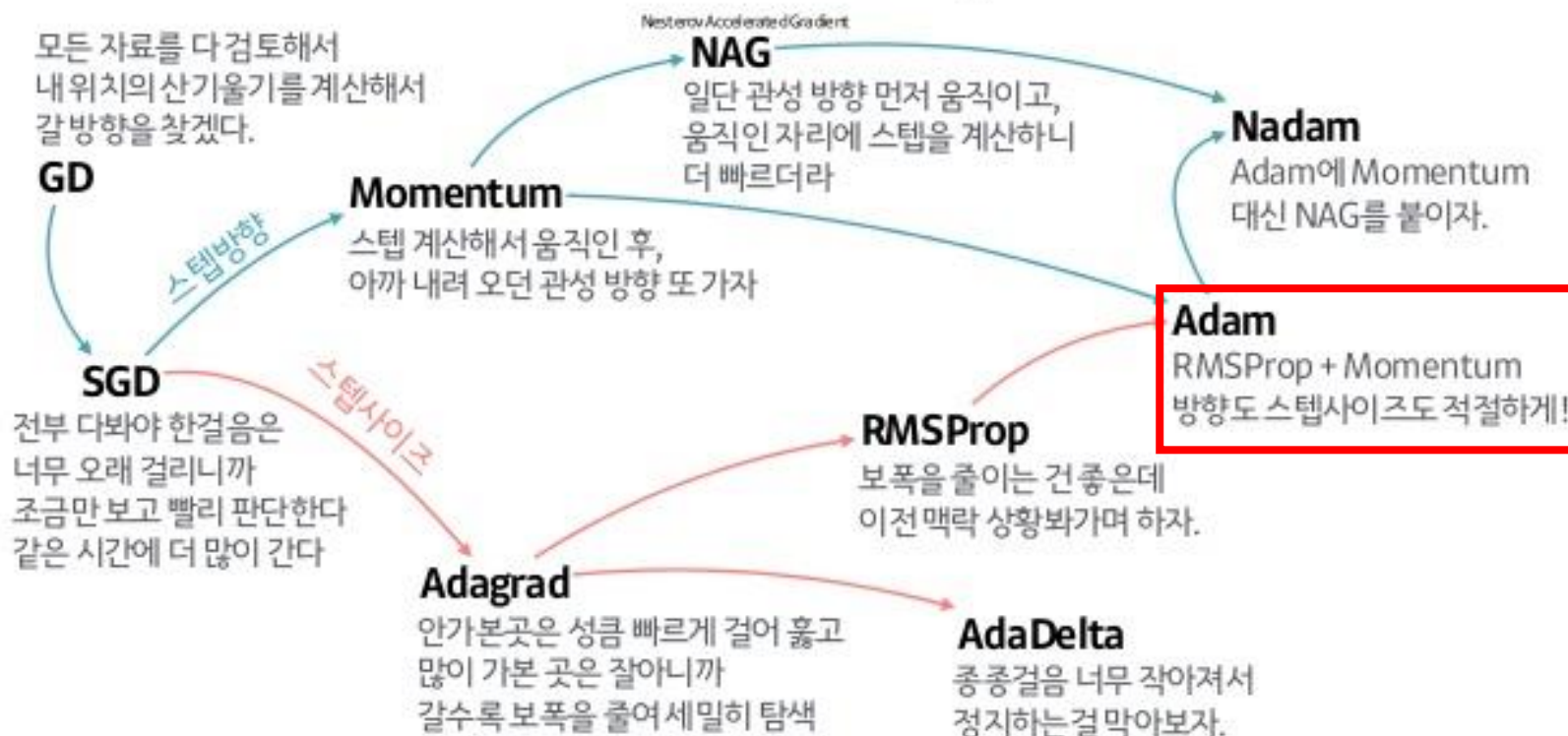
| 알고리즘 | m 이 클 때 | 외부 메모리 학습 자원 | n 이 클 때 | 하이퍼 파라미터 수 | 스케일 조정 필요 | 사이킷런 |
|-------------|-----------|-----------------|-----------|---------------|--------------|------------------|
| 정규방정식 | 빠름 | No | 느림 | 0 | No | N/A |
| SVD | 빠름 | No | 느림 | 0 | No | LinearRegression |
| 배치 경사 하강법 | 느림 | No | 빠름 | 2 | Yes | SGDRegressor |
| 확률적 경사 하강법 | 빠름 | Yes | 빠름 | ≥ 2 | Yes | SGDRegressor |
| 미니배치 경사 하강법 | 빠름 | Yes | 빠름 | ≥ 2 | Yes | SGDRegressor |

그 외 최적화 방법들



그 외 최적화 방법들

산 내려오는 작은 오솔길 찾기(Optimizer)의 발달 계보



선형회귀 수학적 이해

- 선형회귀식 정의하기

$$\hat{y} = b + wX$$

- 제곱 오차 정의하기

$$Residual = (y - \hat{y})^2$$

- 가중치에 대하여 제곱 오차 미분하기

$$\begin{aligned}\frac{\partial Residual}{\partial w} &= \frac{\partial (y - \hat{y})^2}{\partial w} \\ &= 2(y - \hat{y}) \left(-\frac{\partial}{\partial w} \hat{y} \right) \\ &= 2(y - \hat{y})(-x)\end{aligned}$$

- 가중치 업데이트

$$\begin{aligned}w_{n+1} &= w_n - \frac{\partial Residual}{\partial w} \\ w_{n+1} &= w_n + 2(y - \hat{y})(x)\end{aligned}$$

선형회귀 수학적 이해

- 절편에 대하여 제공오차 미분하기

$$\begin{aligned}\frac{\partial \text{Residual}}{\partial b} &= \frac{\partial (y - \hat{y})^2}{\partial b} \\ &= 2(y - \hat{y}) \left(-\frac{\partial}{\partial b} \hat{y} \right) \\ &= 2(y - \hat{y})(-1)\end{aligned}$$

- 절편 업데이트

$$\begin{aligned}b_{n+1} &= b_n - \frac{\partial \text{Residual}}{\partial b} \\ b_{n+1} &= b_n + 2(y - \hat{y})\end{aligned}$$

선형회귀 수학적 이해

● 파이썬 코딩

```
class LinearRegression:

    def __init__(self):
        self.w = 1.0    # initialize weight
        self.b = 1.0    # initialize intercept
        self.lr = 0.01


    def forpass(self, x):
        y_hat = x * self.w + self.b    # calculate y = wx + b
        return y_hat

    def backprop(self, x, err):
        w_grad = (-1) * 2 * x * err    # calculate gradient of weight
        b_grad = (-1) * 2 * 1 * err    # calculate gradient of intercept
        return w_grad, b_grad

    def fit(self, x, y, epochs=5):
        for i in range(epochs):        # repeat depend on epoch
            for x_i, y_i in zip(x, y): # repeat for all samples
                y_hat = self.forpass(x_i) # forward

                err = (y_i - y_hat)      # backpropagation*
                w_grad, b_grad = self.backprop(x_i, err) # calculate backpropagation

                # update
                self.w -= self.lr * w_grad    # weight update
                self.b -= self.lr * b_grad    # intercept update
```



[실습]

[다항 선형 회귀]

다항 회귀

- 간단한 2차방정식으로 비선형 데이터 생성
 - 선형 회귀만으로는 근사가 안됨

```
m = 100  
X = 6 * np.random.rand(m, 1) - 3  
y = 0.5 * X**2 + X + 2 + np.random.randn(m, 1)
```

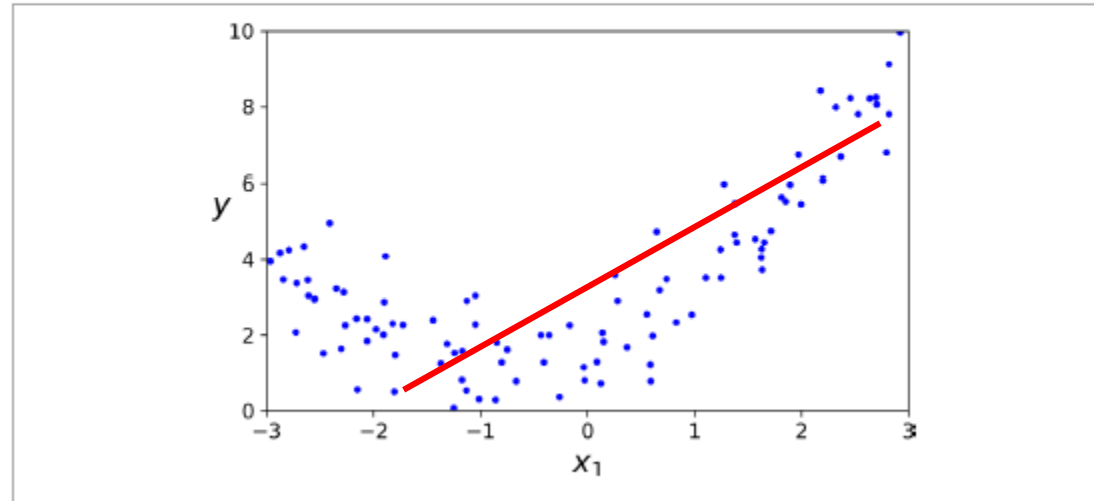


그림 4-12 노이즈가 포함된 비선형 데이터셋

다항 회귀

- 간단한 2차방정식으로 비선형 데이터 생성
 - 특성이 여러 개일 때는 다항 회귀를 사용하자

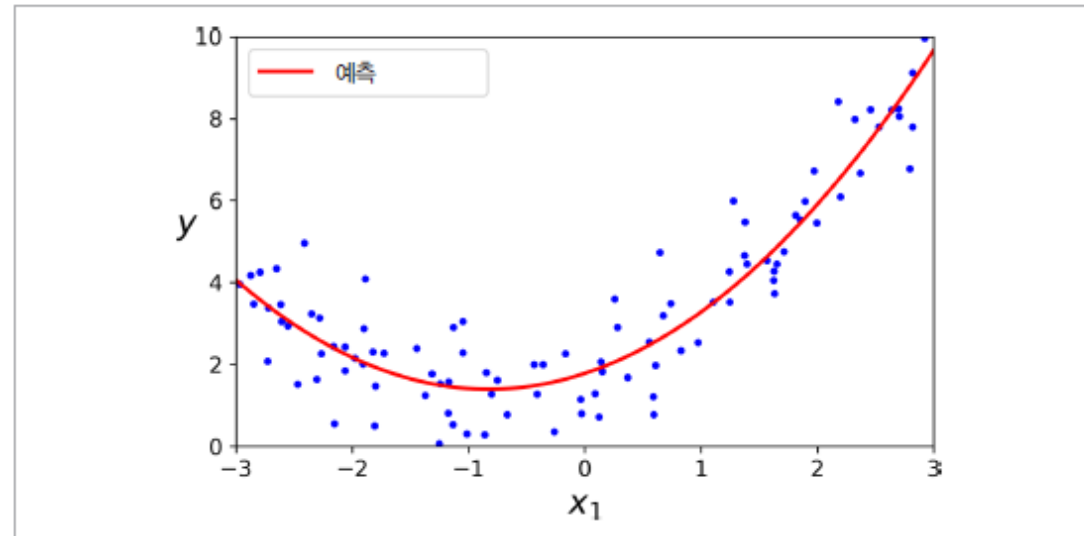


그림 4-13 다항 회귀 모델의 예측

[다중 회귀]

다중 선형 회귀

- 단순선형 회귀에서 독립변수의 개수만 늘어난 선형회귀이다.
- 따라서, 단순선형과 동일한 절차를 이용하여 분석을 수행 할 수 있다.
- 단, 변수의 수가 많아지므로 이로 인해 발생할 수 있는 경우들을 고려하여야 한다.

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \dots + \hat{\beta}_n X_n$$

다중 선형 회귀

● 독립변수들의 최초 선택

- 회귀분석의 목적은 종속변수를 가장 잘 설명하는 독립변수들의 특징을 찾아내어 예측하는 것이다.
- 즉, 독립변수 일부를 임의로 누락시키면 해당 모형의 설명력이 낮아지는 문제를 불러올 수 있다.
- 따라서 통계적으로 회귀분석을 수행 하는 경우 '**다중공선성**'을 고려하여 변수를 설정한다.



다중 선형 회귀

- 다중 공선성(Multi-collinearity)

- 독립변수들 간에 강한 상관관계가 나타나는 것을 다중 공선성 이라 한다.
- 독립변수들 사이의 상관계수 R 이 높으면 회귀모형이 유의미하다고 보기 어렵다.
- 다중공선성문제를 피하려면 상관계수가 높은 변수들을 삭제하거나, 차원축소 기법을 이용하여 의존적인 성분을 제거한 뒤 회귀 분석을 수행해야 한다.

[학습 곡선]

학습 곡선

- 일반화가 잘 되고 있는지 확인하기 위해선?

- 학습 세트와 훈련 세트를 나누고
- 학습 곡선을 살펴보자

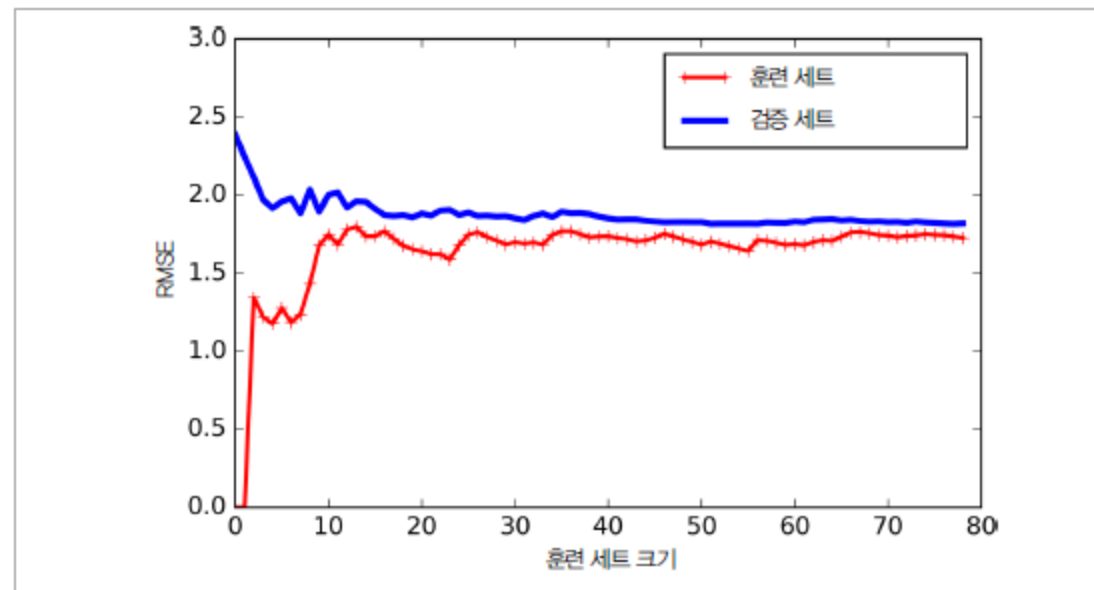


그림 4-15 학습 곡선

학습 곡선

- 모델이 너무 복잡하다면
 - 훈련 데이터의 오차는 작을지 몰라도
 - 이렇게 오버 피팅이 발생함을 확인할 수 있음

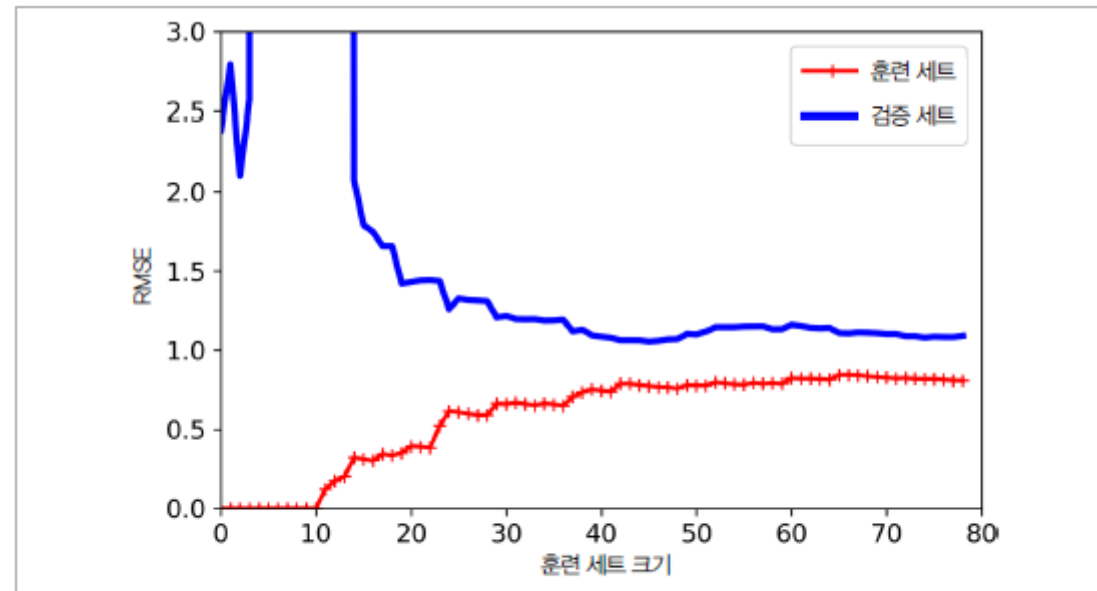


그림 4-16 10차 다항 회귀의 학습 곡선

편향 / 분산 트레이드오프

● 편향

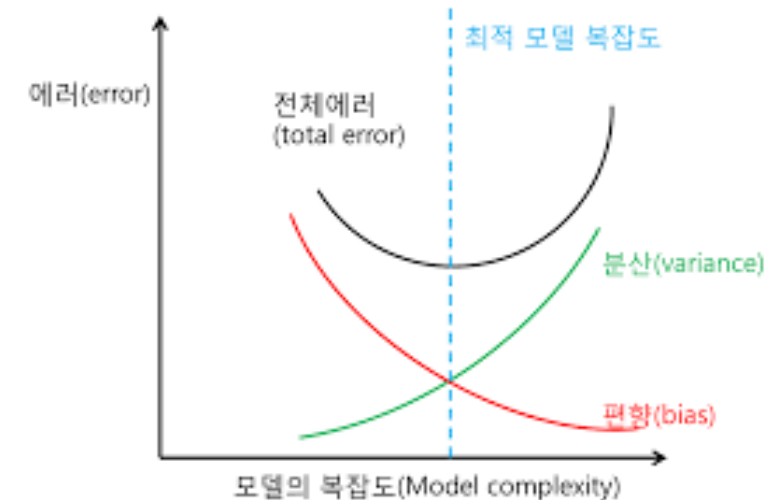
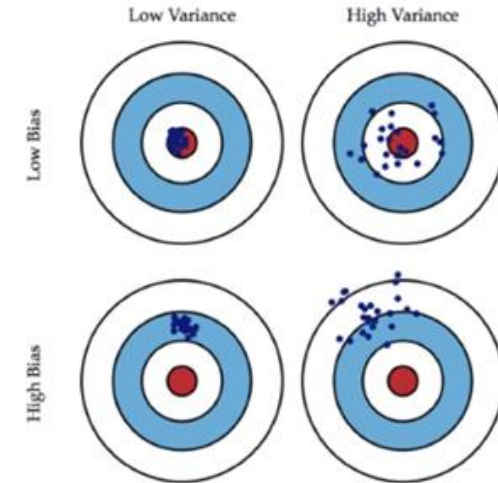
- 일반화 오차 중에서 편향은 잘못된 가정으로 인한 것

● 분산

- 훈련 데이터에 있는 작은 변동에 모델이 과도하게 민감하기 때문에 발생
- 자유도가 높은 모델이 높은 분산을 가지기 쉽기 때문에 오버피팅이 발생할 확률 높음

● 편향과 분산을 동시에 줄일 수 없다

- 데이터 자체에 있는 노이즈 때문
- 노이즈를 제거하면 가능할 수 있음



[규제가 있는 선형 모델]

규제가 있는 선형 회귀

● 회귀모형과 회귀계수

- 지금까지의 선형 회귀 모형은 비용함수를 최소화 하는 것에 초점을 맞추었다.
- 이에 따라서 훈련데이터는 지나치게 잘 맞추고, 회귀계수가 커지게 되어, 평가 데이터에서는 올바르게 예측하지 못할 수 있다. (overfitting)
- 비용함수는 최소화 하면서도 회귀계수값이 너무 커지지 않도록 균형을 이룰 필요가 있다.
- 결과적으로 아래와 같이 비용함수가 수정된다.

$$Loss = \underbrace{\sum (y - \hat{y})^2}_{\text{원래 비용 함수}} + \underbrace{\alpha * R(w)}_{\text{규제항}}$$

규제가 있는 선형 회귀

● 규제항

- 매개변수 α 값에 따라 규제항의 비중이 달라짐
- 만약 α 가 0 이라면, 비용함수는 기존과 동일 하게 된다.
- 만약 α 가 매우 큰 값이라면, 상대적으로 $R(w)$ 값이 비용함수의 대부분을 차지하게 되어 $R(w)$ 를 최소화 시키는 것이 된다.

$$Loss = \underbrace{\sum (y - \hat{y})^2}_{\text{원래 비용 함수}} + \underbrace{\alpha * R(w)}_{\text{규제항}}$$

규제가 있는 선형 회귀

● 릿지 회귀

- 규제항 $\alpha \sum_{i=1}^n \theta_i^2$ 이 비용 함수에 추가
- 모델의 가중치가 가능한 한 작게 유지되도록 노력 → 아예 0이 되진 않음
- 훈련 시에만 비용함수에 추가, 평가 시엔 사용하지 않음
- α 값에 따라서 규제 정도가 달라짐
- L2 규제라고도 함

```
>>> from sklearn.linear_model import Ridge
>>> ridge_reg = Ridge(alpha=1, solver="cholesky")
>>> ridge_reg.fit(X, y)
>>> ridge_reg.predict([[1.5]])
array([[1.55071465]])
```

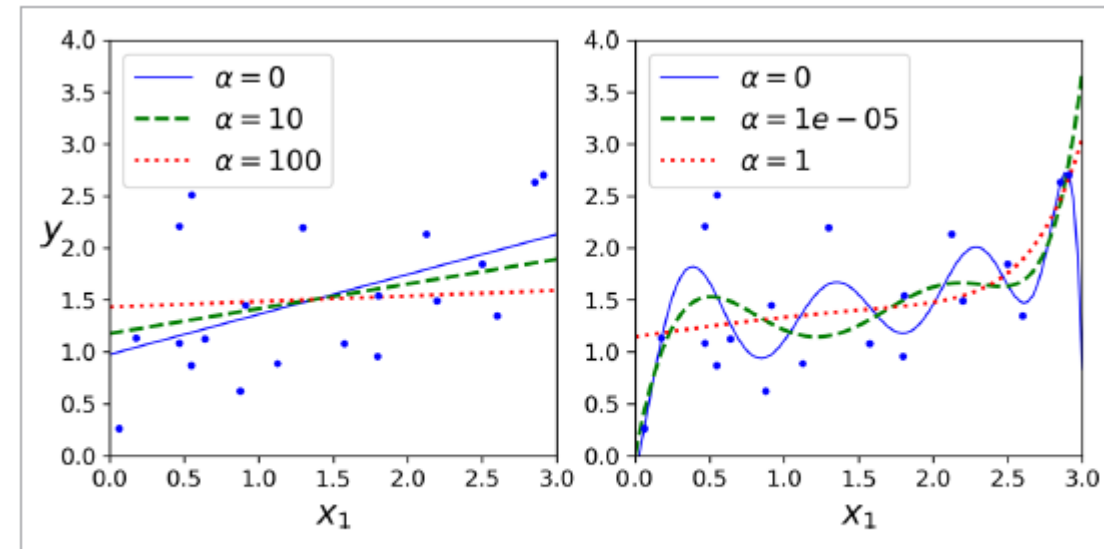


그림 4-17 다양한 수준의 릿지 규제를 사용한 선형 회귀(왼쪽)와 다항 회귀(오른쪽).

규제가 있는 선형 회귀

● 라쏘 회귀

- 규제항 $\alpha \sum_{i=1}^n |\theta_i|$ 이 비용 함수에 추가
- 덜 중요한 특성의 가중치는 점점 제거됨
- 자동으로 특성 선택되는 역할
- α 값에 따라서 규제 정도가 달라짐
- L1규제라고도 함

```
>>> from sklearn.linear_model import Lasso
>>> lasso_reg = Lasso(alpha=0.1)
>>> lasso_reg.fit(X, y)
>>> lasso_reg.predict([[1.5]])
array([1.53788174])
```

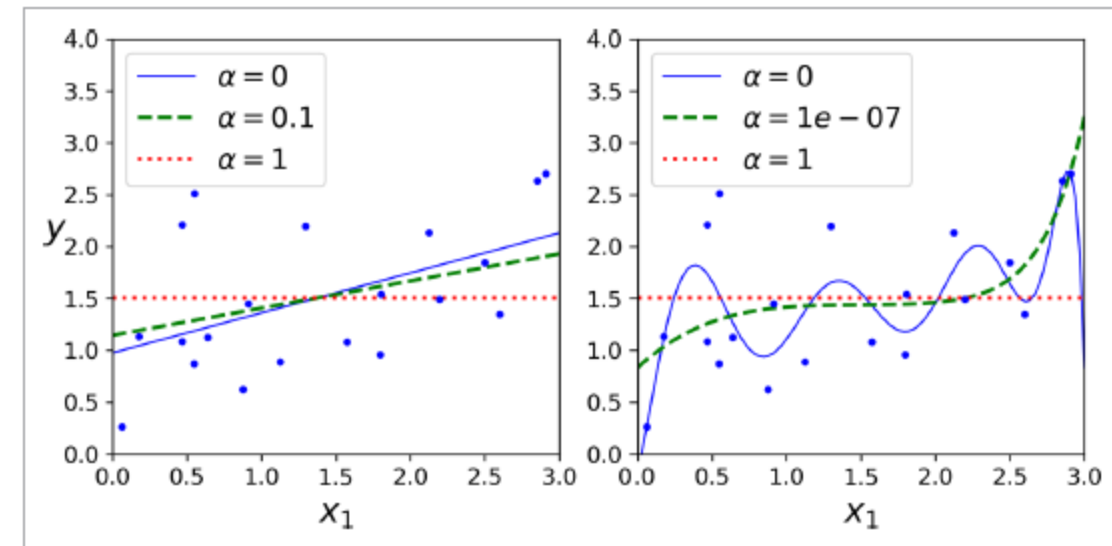


그림 4-18 다양한 수준의 라쏘 규제를 사용한 선형 회귀(왼쪽)와 다항 회귀(오른쪽).

규제가 있는 선형 회귀

- 릿지 회귀 vs 라쏘 회귀

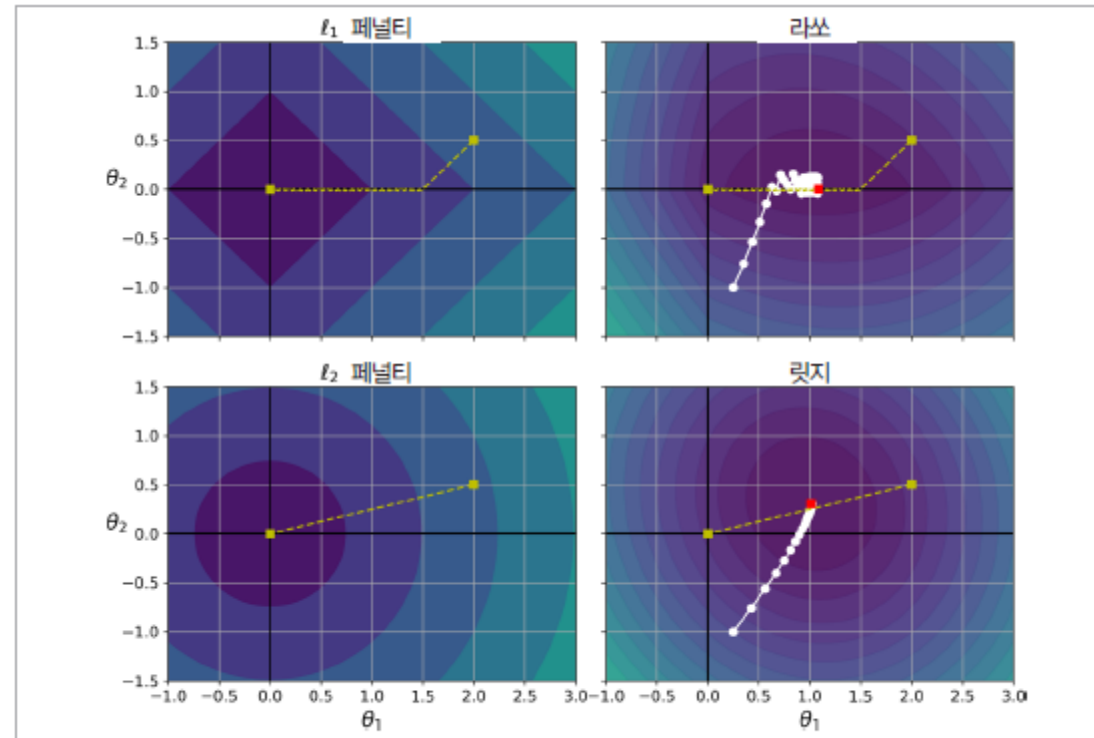


그림 4-19 라쏘 대 릿지 규제

규제가 있는 선형 회귀

● 엘라스틱 회귀

- 릿지와 라소를 혼합한 회귀를 엘라스틱 회귀라고 한다.
- 릿지 규제를 사용해야 할지 라소 규제를 사용해야 할지 모를 때 사용
- 감마는 릿지와 라소를 혼합할 비율을 설정한다.

$$Loss = \underbrace{\sum (y - \hat{y})^2}_{\text{원래 비용 함수}} + \underbrace{(1 - \gamma) * \alpha * \text{릿지}}_{\text{릿지}} + \underbrace{\gamma \text{ 라소}}_{\text{라소}}$$

```
>>> from sklearn.linear_model import ElasticNet
>>> elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5)
>>> elastic_net.fit(X, y)
>>> elastic_net.predict([[1.5]])
array([1.54333232])
```

규제가 있는 선형 회귀

● 조기 종료

- 검증 에러가 최솟값에 도달하면 바로 훈련 중지시킴
- 과대적합되기 전에 훈련을 멈추게 함으로써 규제를 함
- **사이킷런에서는** 에러가 높아지는 지점에서 자동으로 멈추게 되어있다.

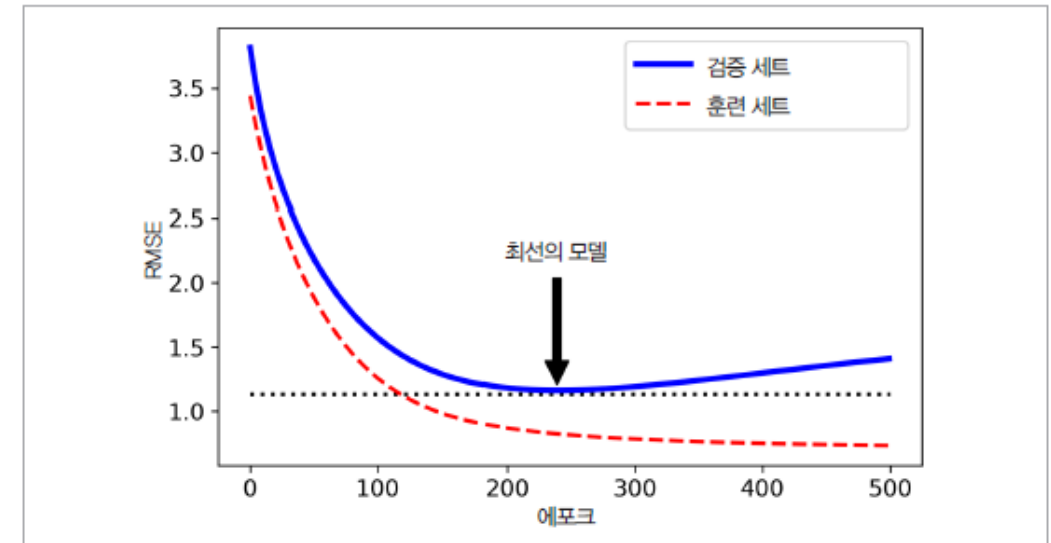


그림 4-20 조기 종료 규제

[로지스틱 회귀]

로지스틱 회귀

- 선형회귀 모형을 '**분류**'에 적용한 기법
- 데이터가 특정 클래스에 속할 확률을 추정한다.
 - Ex1. 이 이메일이 스팸일 확률은 얼마인가?
 - Ex2. 이번 시험에서 합격할 확률은 얼마인가?
- 다른 선형 회귀 모형과는 다르게 종속변수가 수치형이 아니라 범주형이다.
 - Ex1. 스팸메일 or 정상메일
 - Ex2. 합격, 불합격

로지스틱 회귀

- 특정 클래스에 대하여 추정된 확률이 50% 이상이면 해당 클래스에 속하는 것으로 분류
 - 이와 같은 성질을 만족하는 함수(시그모이드)를 사용하여 분류한다.
 - 예측값 t 가 시그모이드 함수에 전달되면 결과값이 계산 되는데 이때 값이 곧 확률이다.

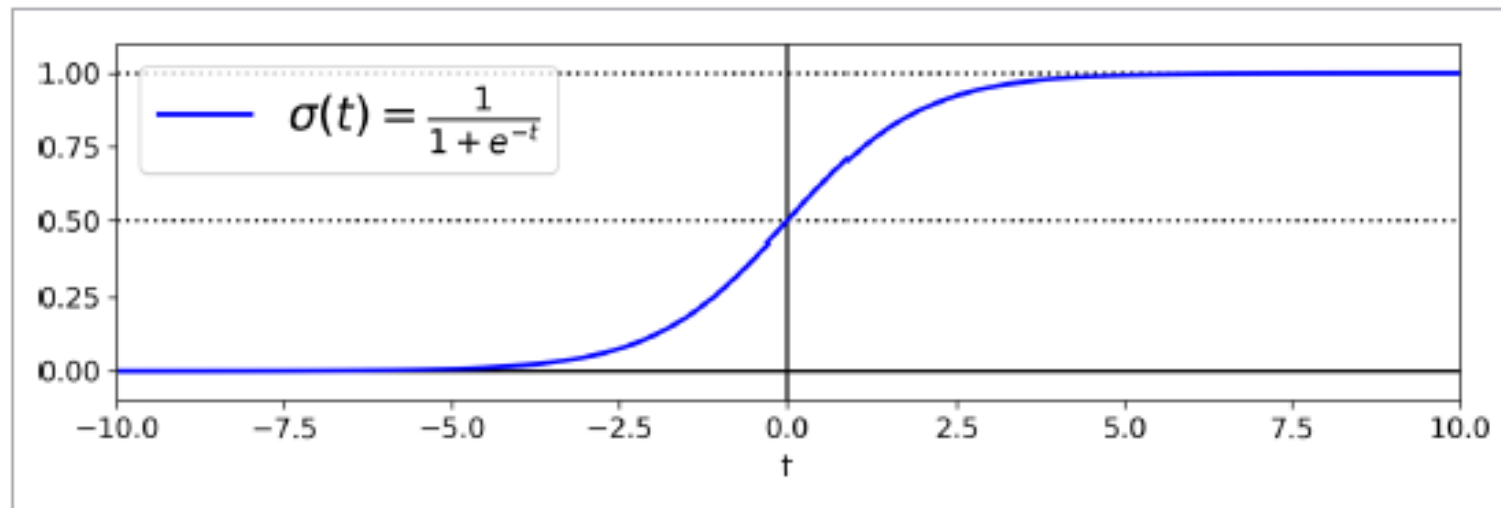


그림 4-21 로지스틱 함수

로지스틱 회귀

● 로그손실

- 로지스틱 회귀의 비용함수
- $J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})]$
 - 양성 샘플을 0에 가까운 확률로 추정하면 비용이 크게 증가
 - 음성 샘플을 1에 가까운 확률로 추정하면 비용이 크게 증가
- 볼록 함수
 - 전역 최솟값을 찾는 것이 보장됨

Diagram illustrating the cross-entropy loss function $D(S, L) = -\sum_i L_i \log(S_i)$.

Two vectors are shown:

- Vector S (Predicted probabilities): $\begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$
- Vector L (Target labels): $\begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \end{bmatrix}$

The loss function is defined as:

$$D(S, L) = -\sum_i L_i \log(S_i)$$

로지스틱 회귀

● 결정 경계

```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> list(iris.keys())
['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename']
>>> X = iris["data"][:, 3:] # 꽃잎의 너비
>>> y = (iris["target"] == 2).astype(np.int) # 1 Iris-Virginica면 1, 그렇지 않으면 0
```

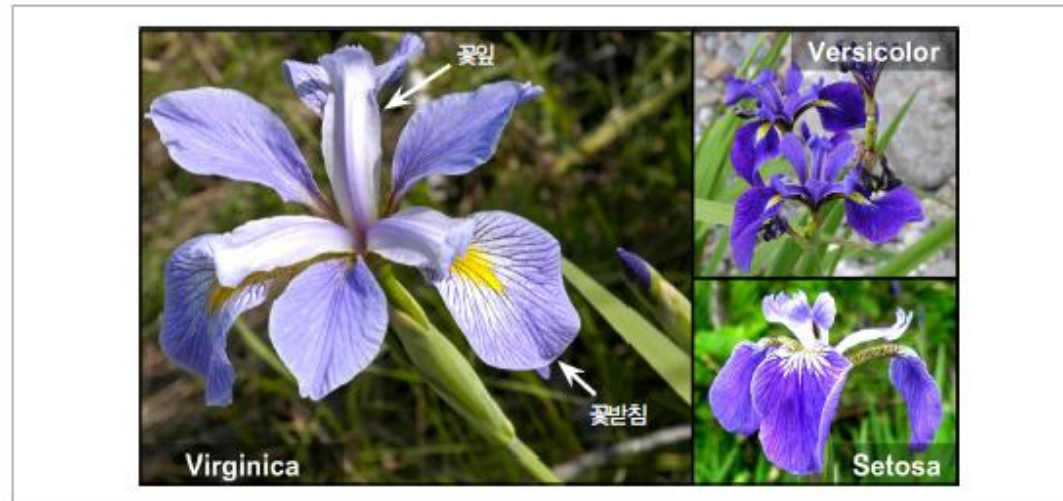


그림 4-22 세 종류의 붓꽃⁴⁰

로지스틱 회귀

- 결정 경계

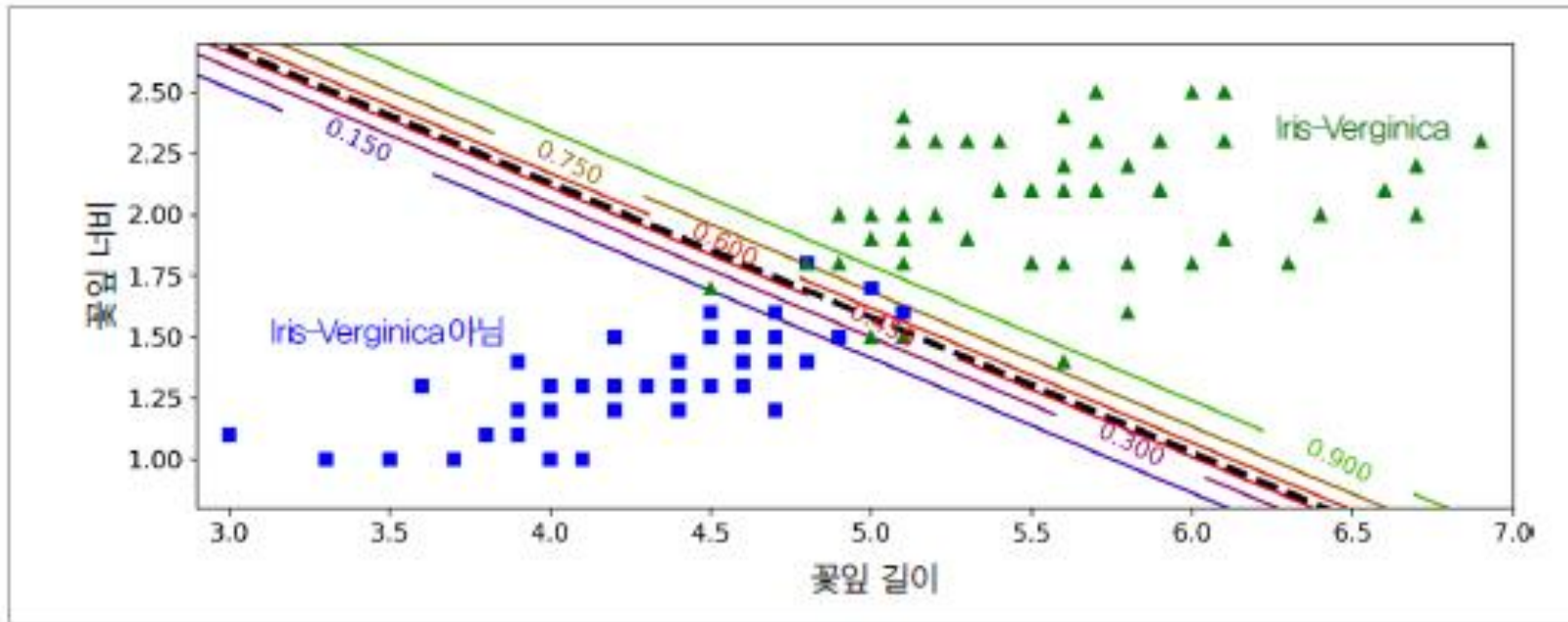



그림 4-24 선형 결정 경계

Quiz

- 선형 회귀 평가지표에는 어떤 것들이 있을까요?
- 결정계수란 무엇일까요?
- 선형회귀를 분류에 사용하기 위해서는 어떠한 함수가 필요한가요?
- 규제는 왜 필요하며 어떤 규제들이 있을까요?

The background is a dark, textured surface featuring a network of glowing red and orange lines that form a complex, interconnected web. Scattered throughout are various geometric shapes, including triangles and polygons, some of which are highlighted in a bright yellow-orange glow. Faint, glowing binary code (0s and 1s) is visible in the lower-left and lower-right areas, adding to the digital or technological theme.

[실습]

붓꽃 데이터셋으로 분류 문제를 풀어봅시다!

- 다음 시간에 -