

8주차(2/3)

아달라인 경사하강법 적용

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

1. 아달라인 경사하강법 적용

■ 학습 목표

- 붓꽃 학습자료의 속성들을 학습한다.
- 붓꽃 학습자료를 바탕으로 아달라인 객체를 테스트한다.
- 모멘텀을 이용하여 비용함수의 값이 최소값으로 수렴하도록 한다.

■ 학습 내용

- 붓꽃 학습자료 속성
- 붓꽃 학습자료 예제
- 지역 최소와 전역 최소
- 모멘텀

1. 붓꽃 학습자료 : 개요

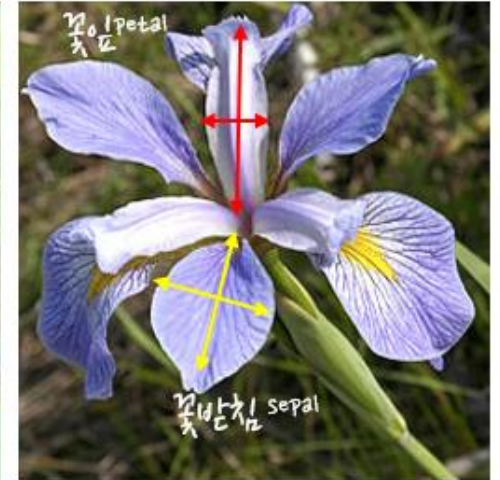
- 1936년 Ronald Fisher 논문에서 사용
- 세가지 종류의 붓꽃 : 세토사, 버시칼라, 버지니카
- 특성
 - 꽃잎의 길이, 너비
 - 꽃받침의 길이, 너비
 - 붓꽃 종류의 이름



세토사 Setosa



버시칼라 Versicolor



버지니카 Virginica

1. 붓꽃 학습자료 : 기계학습을 위한 표기

- 특성 행렬 X 로 표기

- 행 : 샘플 수
- 열 : 특성 수

$$X \in \mathbb{R}^{150 \times 4}$$

$$X = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{pmatrix}$$

1. 붓꽃 학습자료 : 기계학습을 위한 표기

$$x_j^{(i)} = (x_1^{(i)} \quad x_2^{(i)} \quad x_3^{(i)} \quad x_4^{(i)})$$

$$x_j^{(i)} = \begin{pmatrix} x_j^{(1)} \\ x_j^{(2)} \\ \vdots \\ x_j^{(150)} \end{pmatrix}$$

1. 붓꽃 학습자료 : 기계학습을 위한 표기

- 특성 행렬 **X**로 표기
 - **(n x m)** 혹은 **(m x n)**으로 표기 가능

형상 (샘플의 수 **x** 특성의 수)

$$X \in \mathbb{R}^{150 \times 4}$$

$$X = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(150)} & x_2^{(150)} & x_3^{(150)} & x_4^{(150)} \end{pmatrix}$$

형상 (특성의 수 **x** 샘플의 수)

$$X \in \mathbb{R}^{4 \times 150}$$


$$X = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(150)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(150)} \\ x_3^{(1)} & x_3^{(2)} & \cdots & x_3^{(150)} \\ x_4^{(1)} & x_2^{(2)} & \cdots & x_4^{(150)} \end{pmatrix}$$

1. 붓꽃 학습자료 : 기계학습을 위한 표기

$$y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(150)} \end{pmatrix}$$

$y \in \text{big}(\textit{Setosa}, \textit{Vericolor}, \textit{Virginica})$


1. 붓꽃 학습자료: 자료 읽기



```
import pandas as pd
df = pd.read_csv('https://archive.ics.uci.edu/
                 'ml/machine-learning-databases/'
                 'iris/iris.data',
                 header=None)
df.head()
```


1. 붓꽃 학습자료: 자료 읽기

```
import pandas as pd
df = pd.read_csv('https://archive.ics.uci.edu/'
                 'ml/machine-learning-databases/'
                 'iris/iris.data',
                 header=None)
```




`df.head()`

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

2. 붓꽃 데이터 학습 예제 : 자료 읽기

```
import joy
```




```
X, y = joy.iris_data()
```

```
ada = AdalineGD(epochs=10, eta=0.1)
```

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

```
joy.plot_xyw(X, y, ada.w)
```

2. 붓꽃 데이터 학습 예제 : 아달라인 객체 생성 및 학습

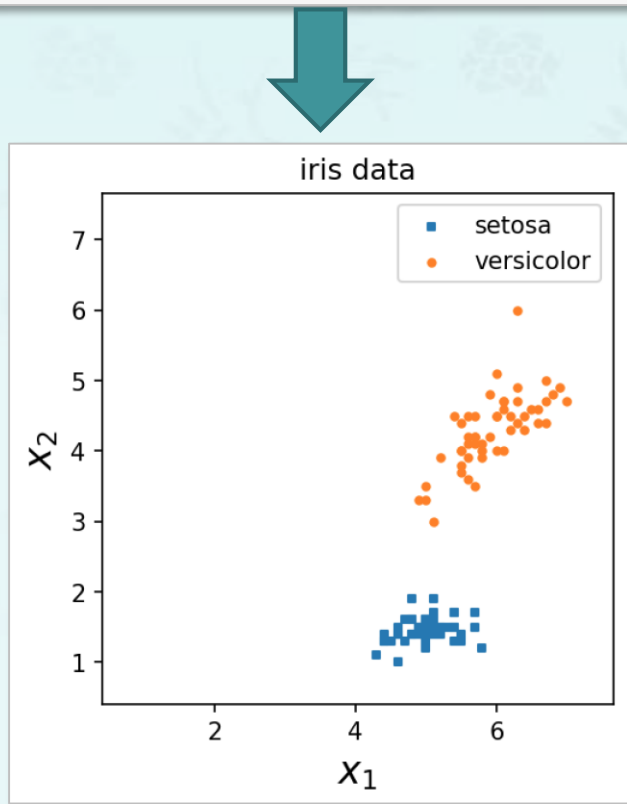


```
import joy

X, y = joy.iris_data()
ada = AdalineGD(epochs=10, eta=0.1)
ada.fit(X, y)
joy.plot_xyw(X, y, ada.w)
```

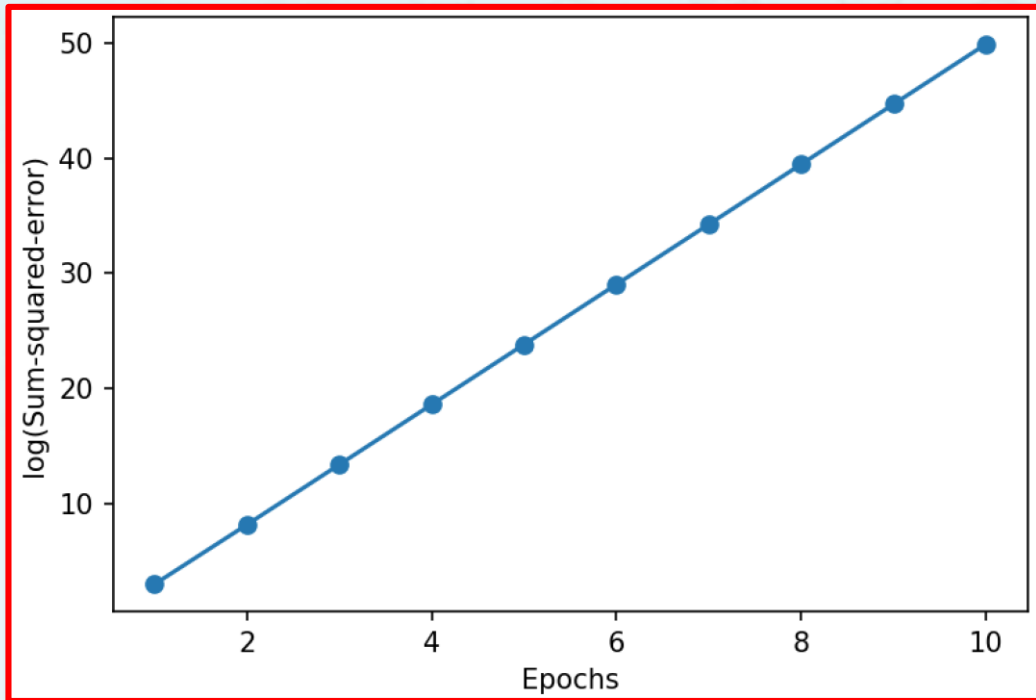
2. 붓꽃 데이터 학습 예제 : 아달라인 객체 생성 및 학습

```
import joy  
  
X, y = joy.iris_data()  
ada = AdalineGD(epochs=10, eta=0.1)  
ada.fit(X, y)  
joy.plot_xyw(X, y, ada.w)
```



2. 붓꽃 데이터 학습 예제 : 비용함수 값 확인

```
plt.plot(range(1, len(ada.cost_) + 1),  
         np.log10(ada.cost_), marker='o')  
plt.xlabel('Epochs')  
plt.ylabel('log(Sum-squared-error)')
```

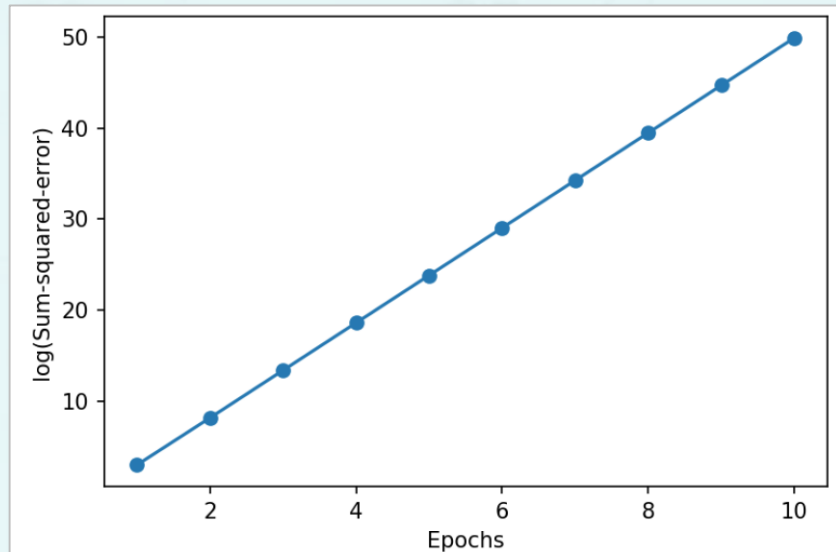


2. 붓꽃 데이터 학습 예제 : 학습률 조정

■ 학습률 ($\eta : \uparrow$)

↓

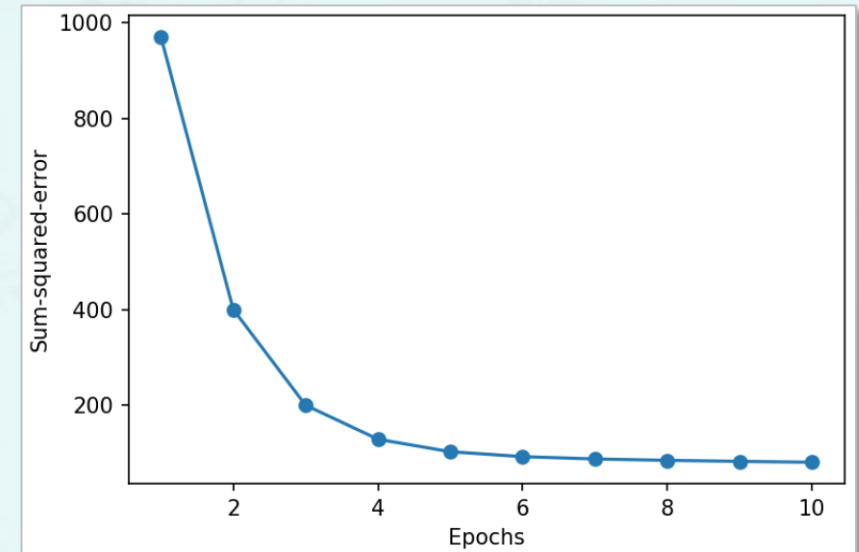
```
X, y = joy.iris_data()
ada1 = AdalineGD(epochs=10, eta=0.1).fit(X,y)
plt.plot(range(1, len(ada1.cost_) + 1), np.log10(ada1.cost_),
         marker='o')
plt.xlabel('Epochs')
plt.ylabel('log(Sum-squared-error)')
plt.show()
```



■ 학습률 ($\eta : \downarrow$)

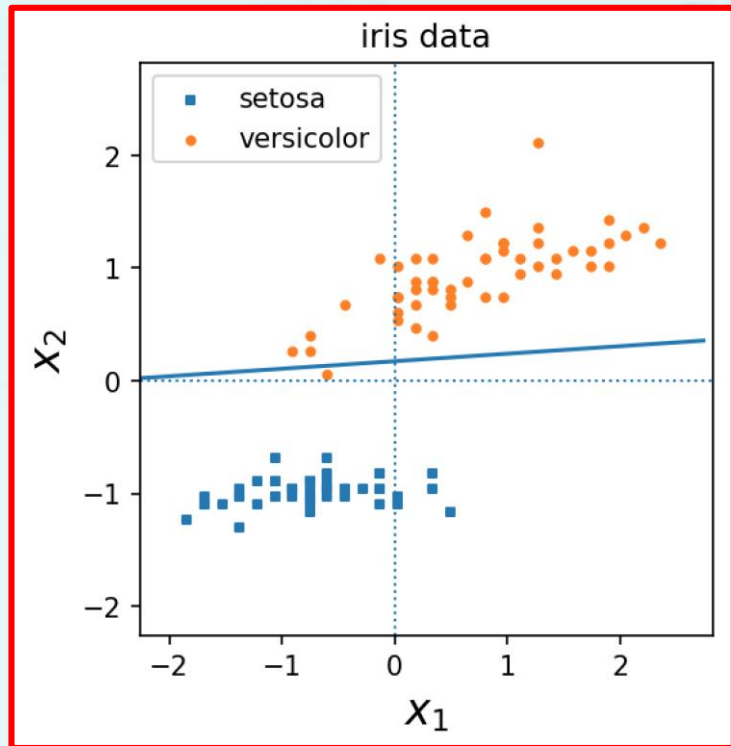
↓

```
X, y = joy.iris_data()
ada2 = AdalineGD(epochs=10, eta=0.0001).fit(X, y)
plt.plot(range(1, len(ada2.cost_) + 1), ada2.cost_,
         marker='o')
plt.xlabel('Epochs')
plt.ylabel('Sum-squared-error')
plt.show()
```



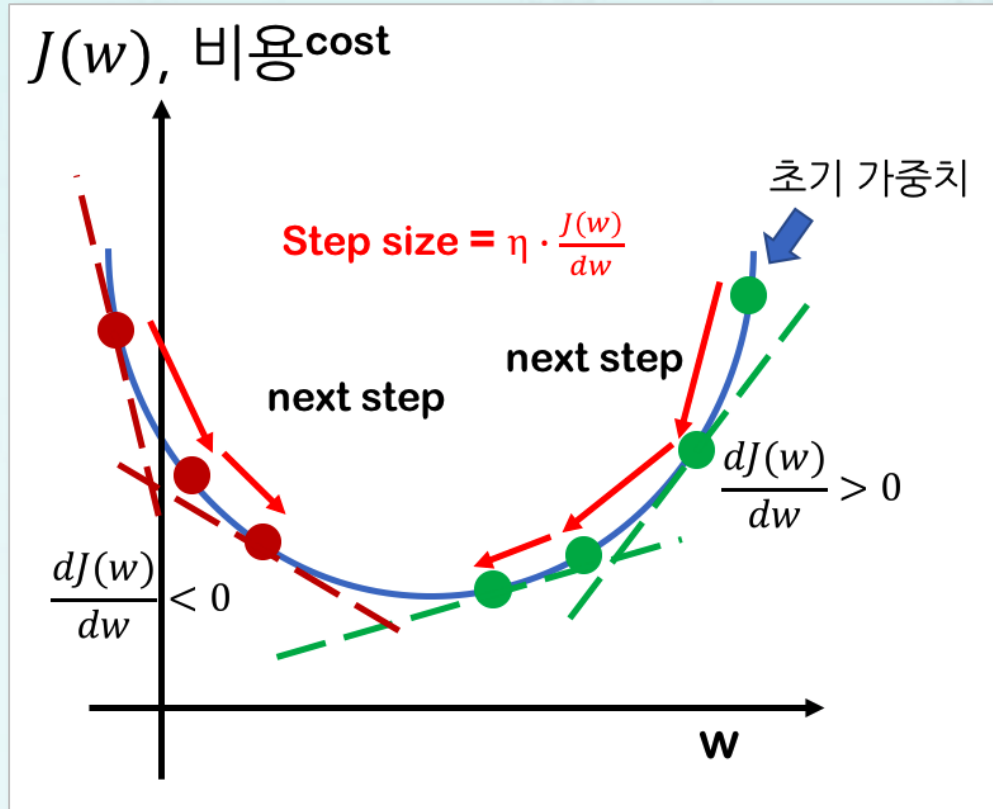
2. 붓꽃 데이터 학습 예제 : 붓꽃자료의 전처리 (표준화)

```
import joy
Xstd, y = joy.iris_data(standardized=True)
ada = AdalineGD(epochs=10, eta=0.001)
ada.fit(Xstd, y)
joy.plot_xyw(Xstd, y, ada.w)
```



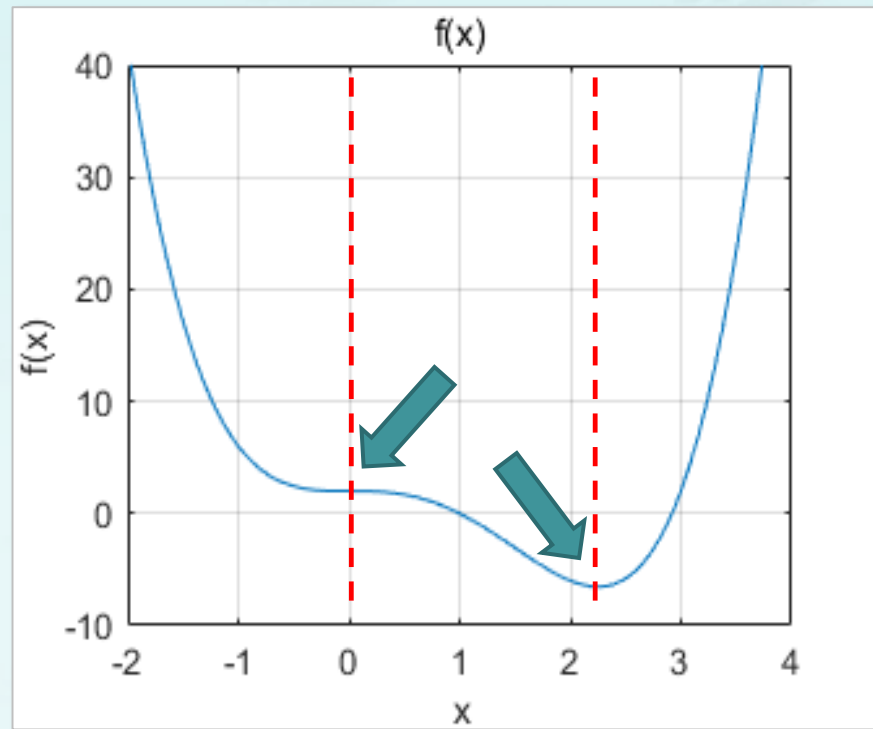
3. 경사하강법: 2차 함수 학습

- 간단한 특성을 가진 자료 경우 가능 (예: 붓꽃 자료)
- 복잡한 특성을 가진 자료 경우 한계 있음

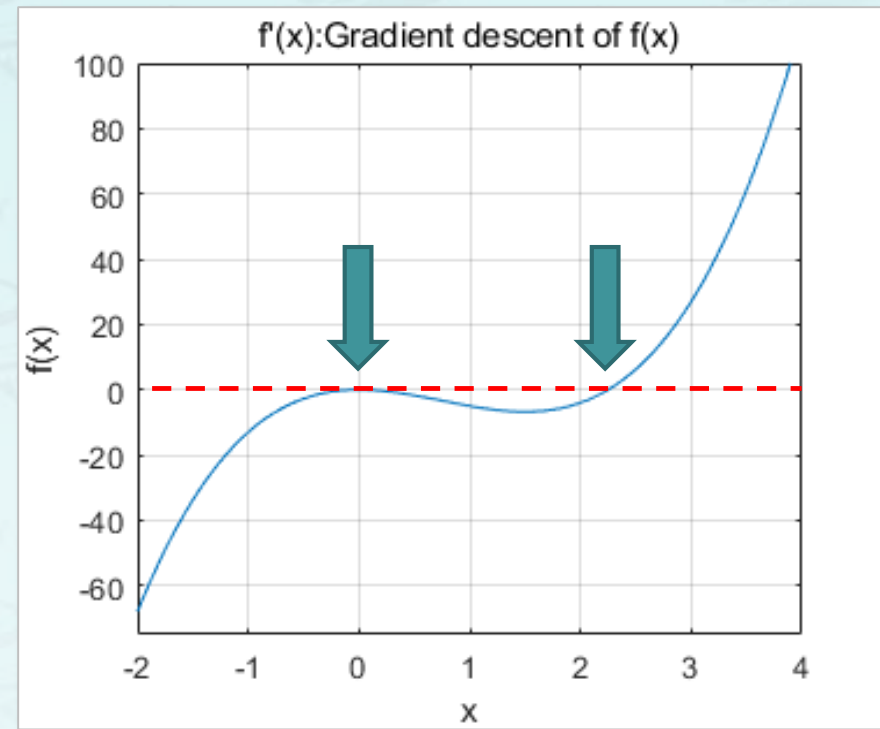


3. 경사하강법: 지역 최소와 전역 최소

- $f(x) = x^4 - 3x^3 + 2$



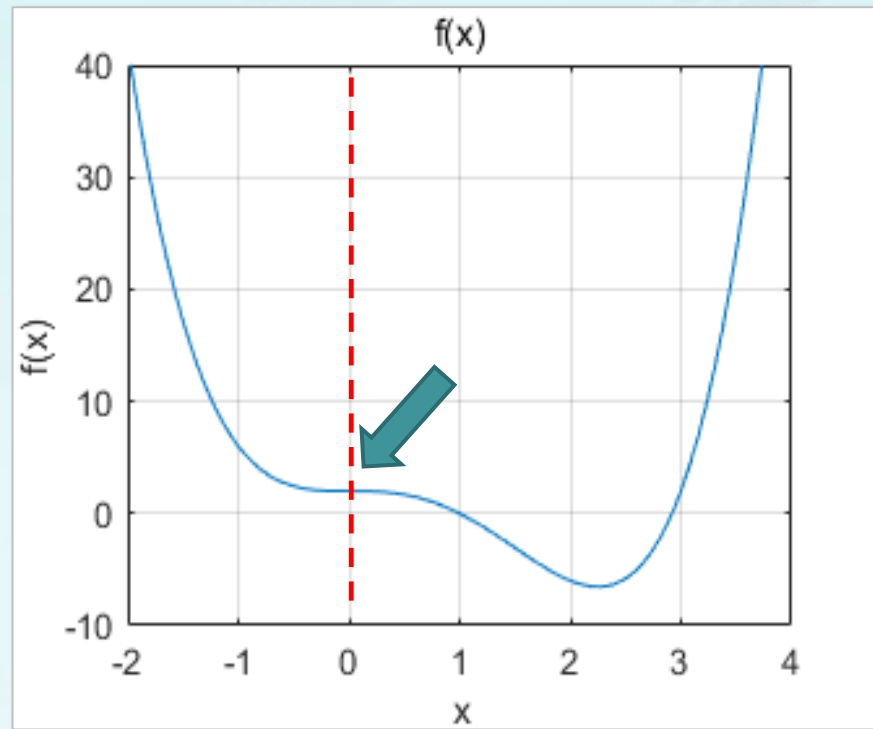
- $f'(x) = 4x^3 - 9x^2$



3. 경사하강법: 지역 최소와 전역 최소

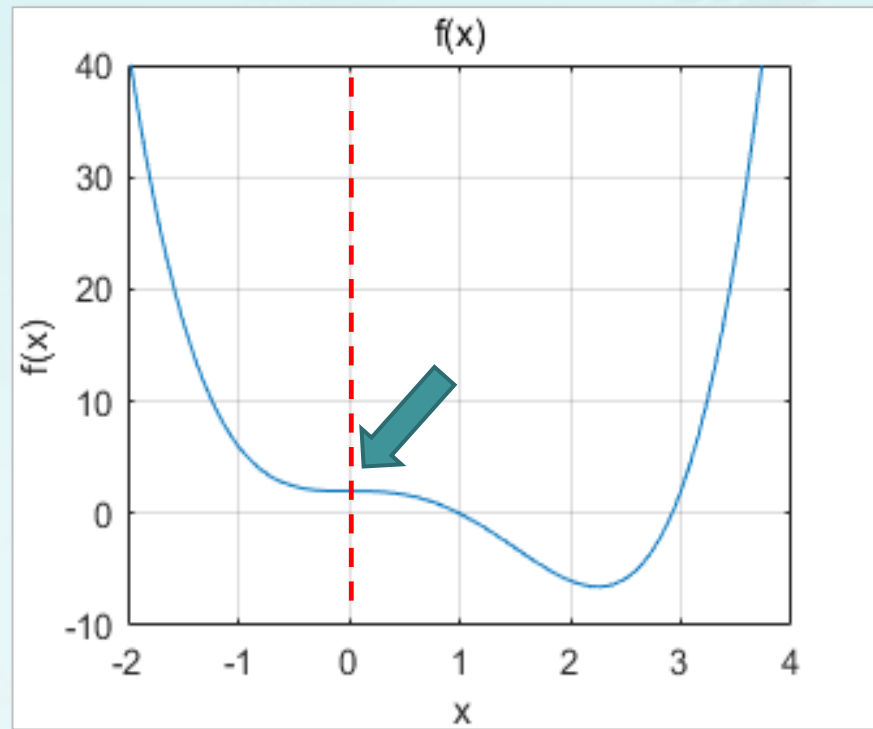
- 지역 최소(Local Minimum)

- $f(x) = x^4 - 3x^3 + 2$



3. 경사하강법: 지역 최소와 전역 최소

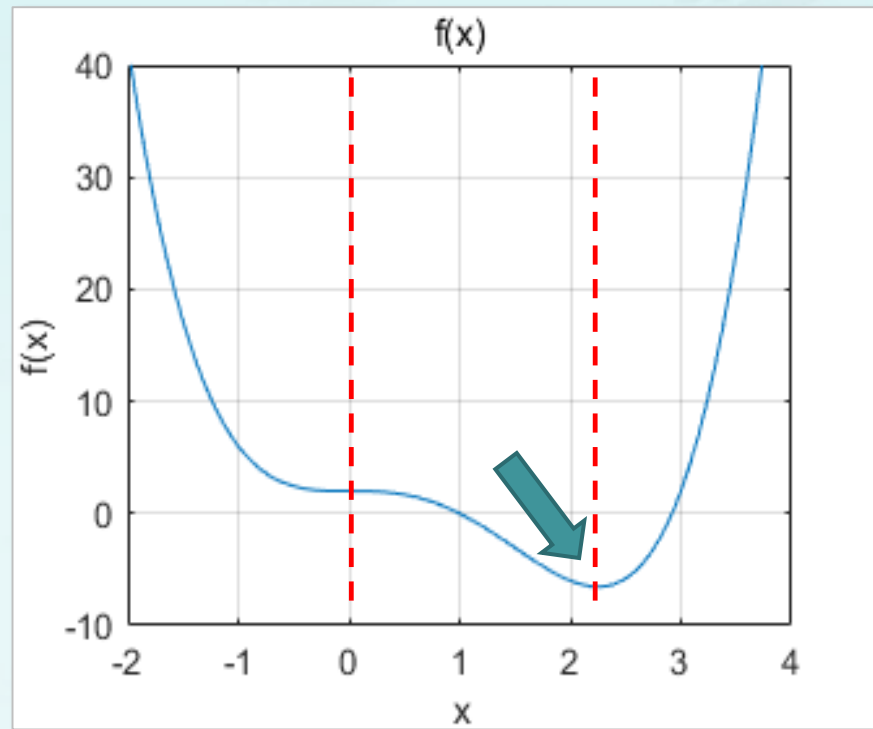
- $f(x) = x^4 - 3x^3 + 2$



- 지역 최소(Local Minimum)
 - 안장점(saddle point)

3. 경사하강법: 지역 최소와 전역 최소

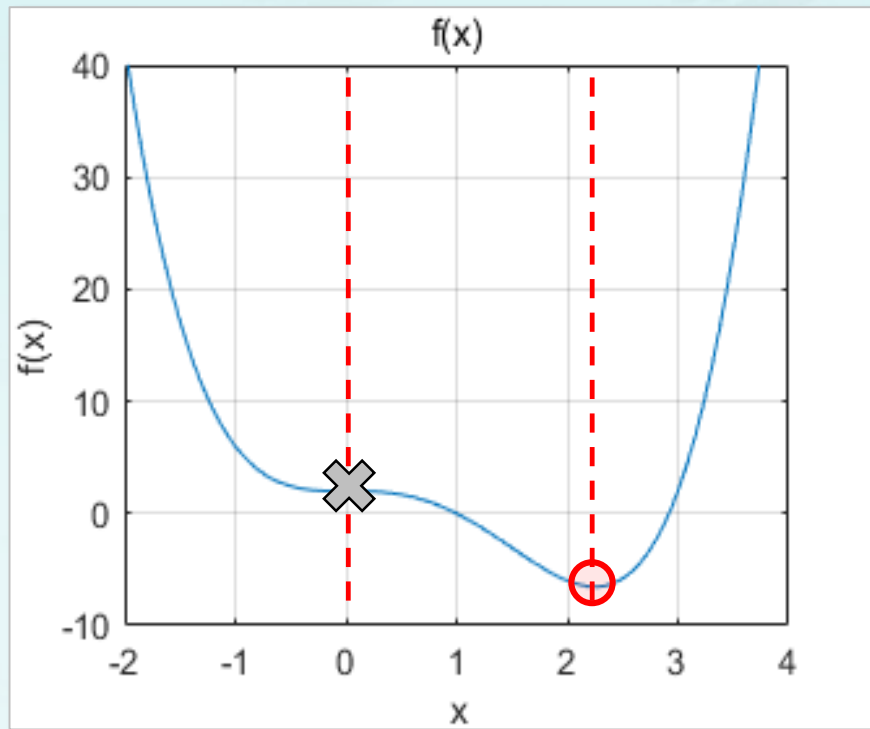
- $f(x) = x^4 - 3x^3 + 2$



- 지역 최소(Local Minimum)
 - 안장점(saddle point)
- 전역 최소(Global Minimum)

3. 경사하강법: 지역 최소와 전역 최소

- $f(x) = x^4 - 3x^3 + 2$

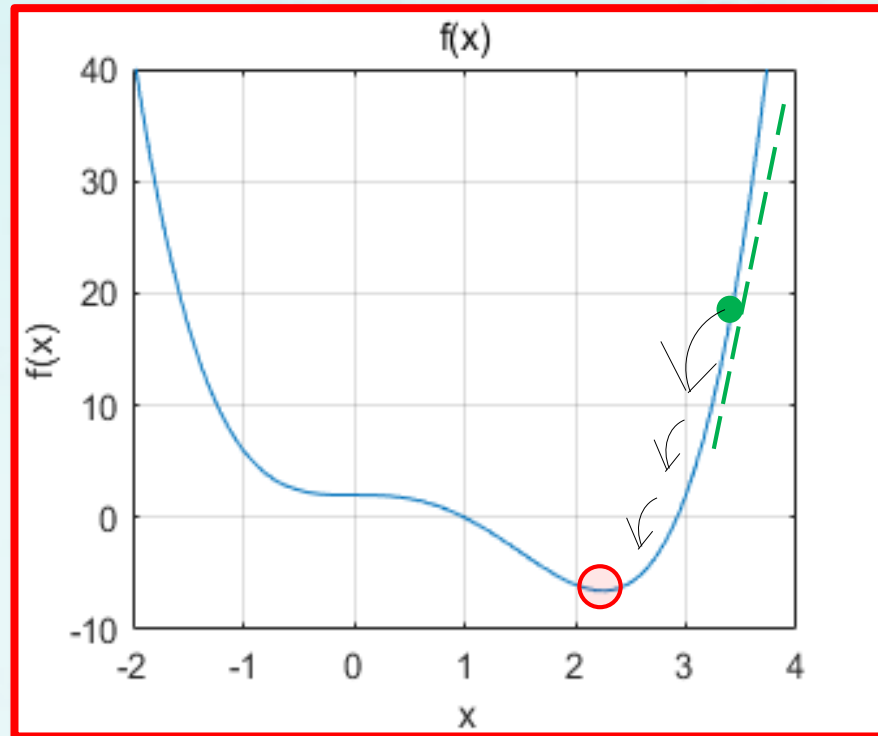


- 지역 최소(Local Minimum)
 - 안장점(saddle point)

- 전역 최소(Global Minimum)

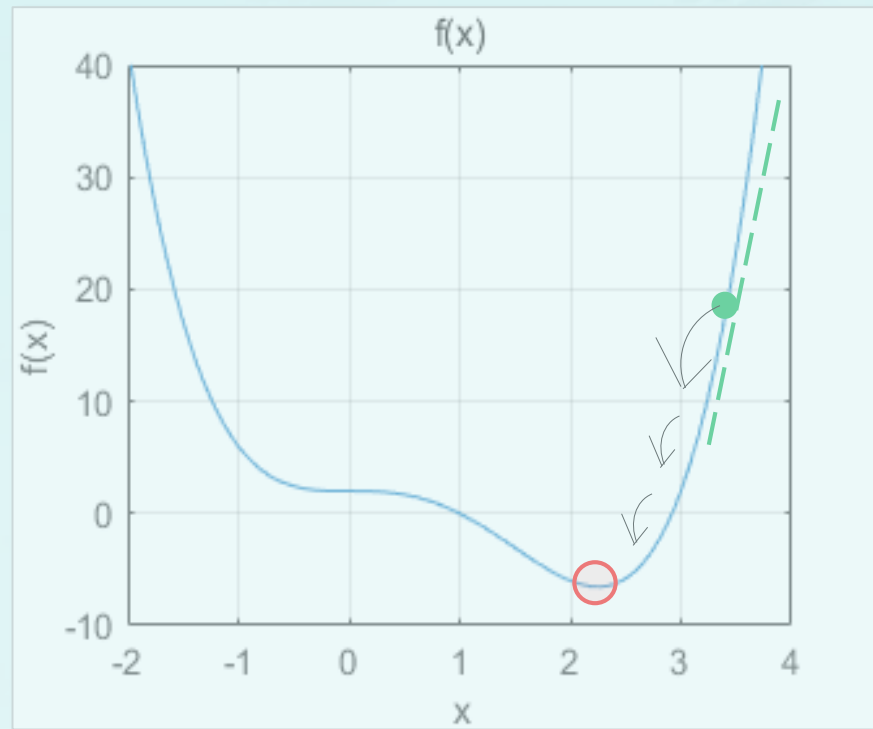
3. 경사하강법: 지역 최소와 전역 최소

- 전역 최소(Global Minimum)

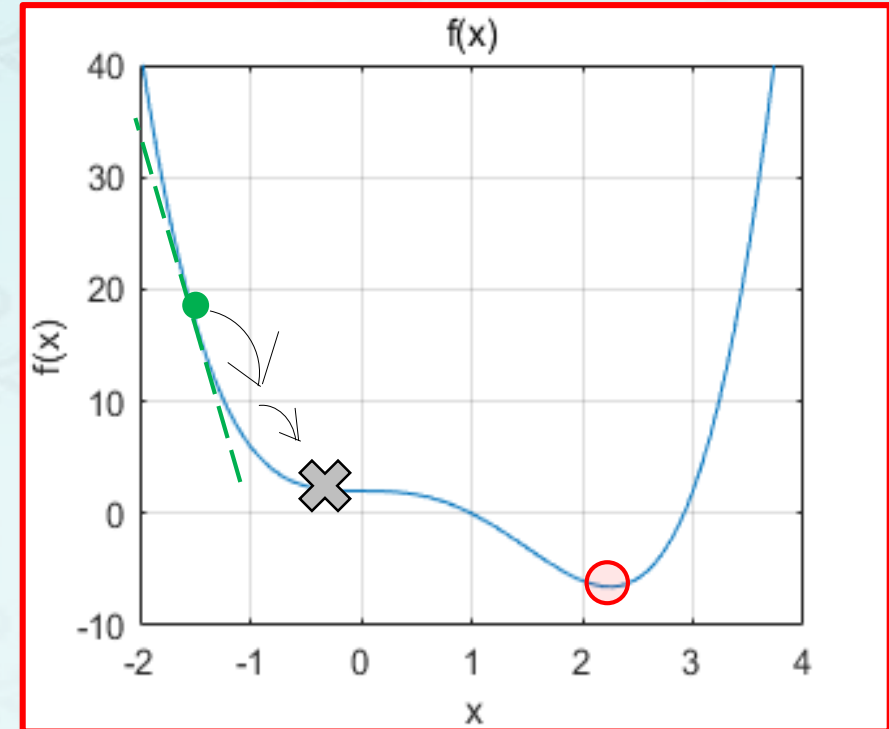


3. 경사하강법: 지역 최소와 전역 최소

- 전역 최소(Global Minimum)

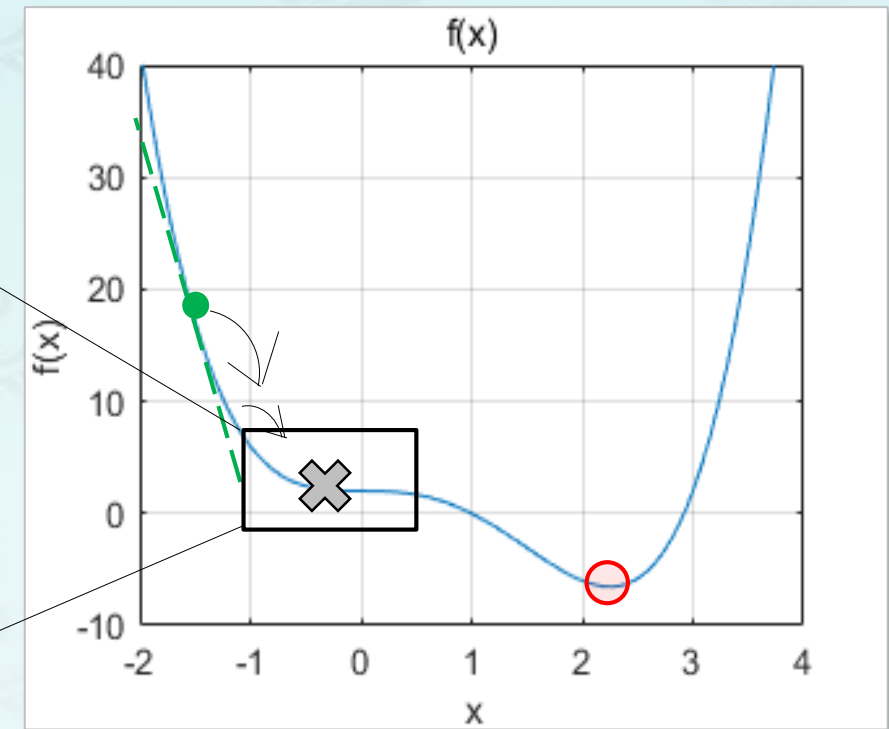
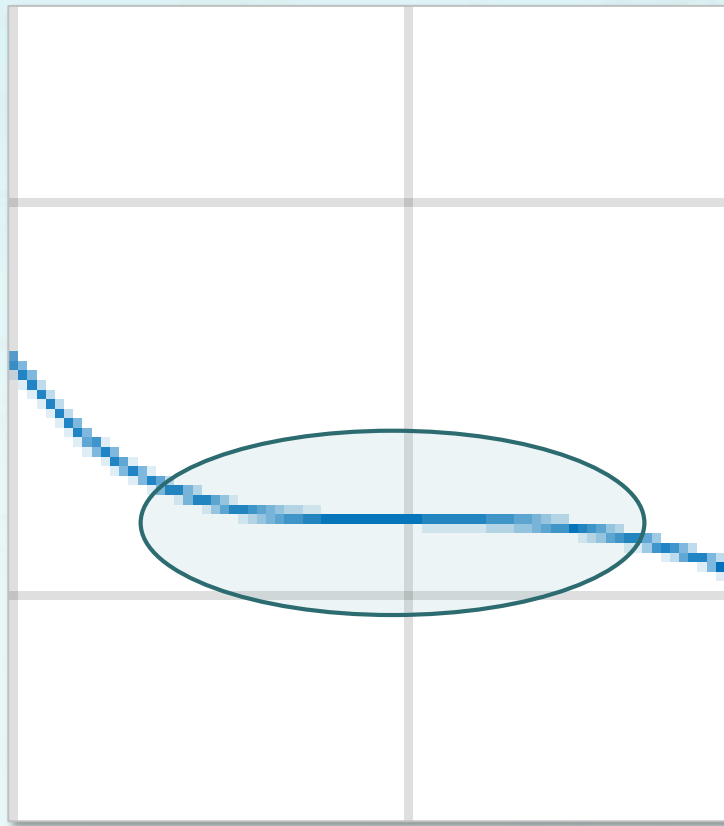


- 지역 최소(Local Minimum)



3. 경사하강법: 지역 최소와 전역 최소

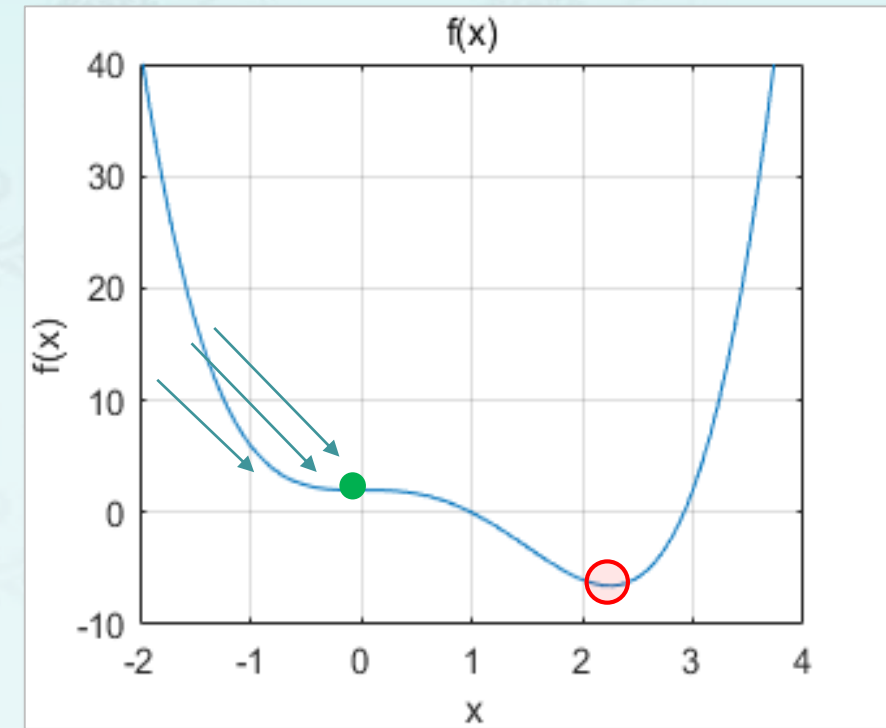
- 지역 최소(Local Minimum)



4. 모멘텀: 개요

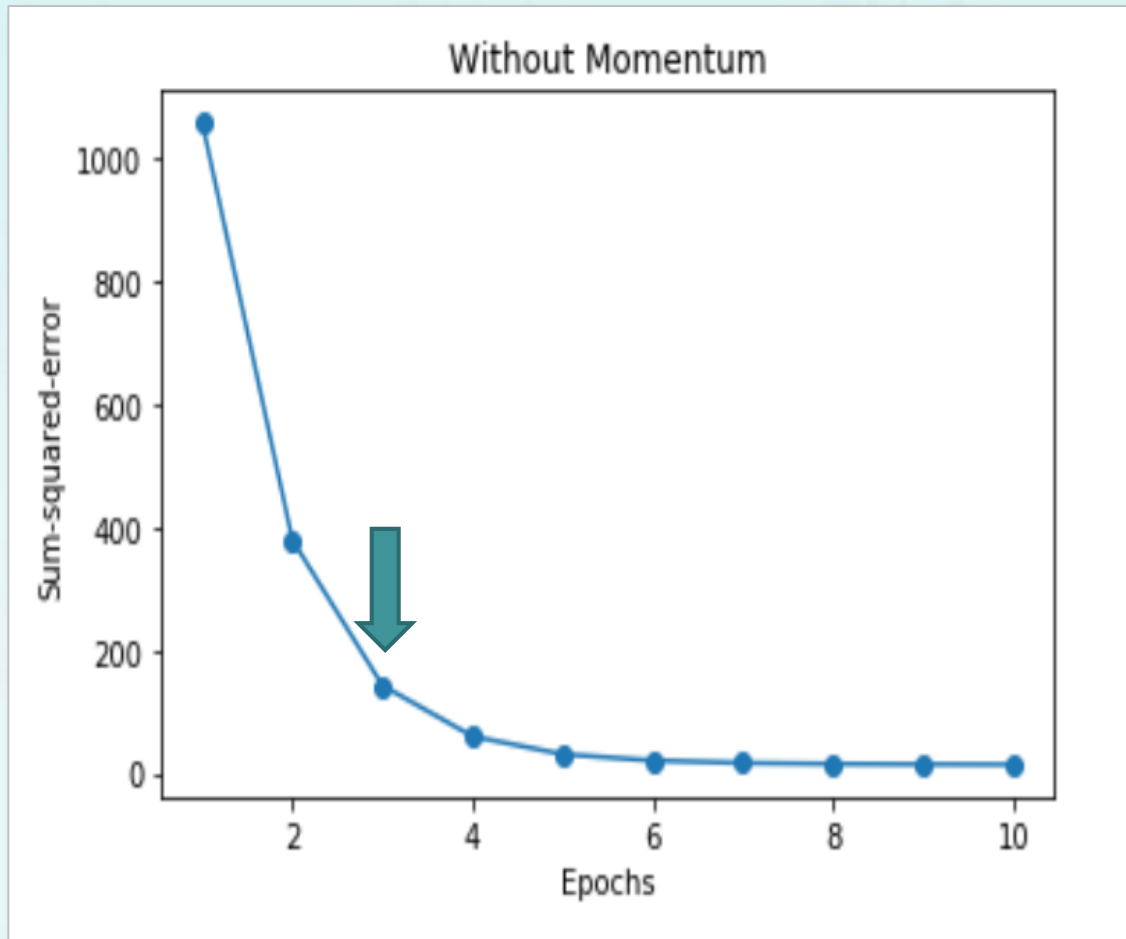
- 운동량 (관성)
- 내려가던 스텝의 방향으로 스텝의 방향을 보정하려는 것

- 모멘텀으로 인해 지역 최소 값에서 전역 최소 값으로 이동할 기회를 갖게 됨

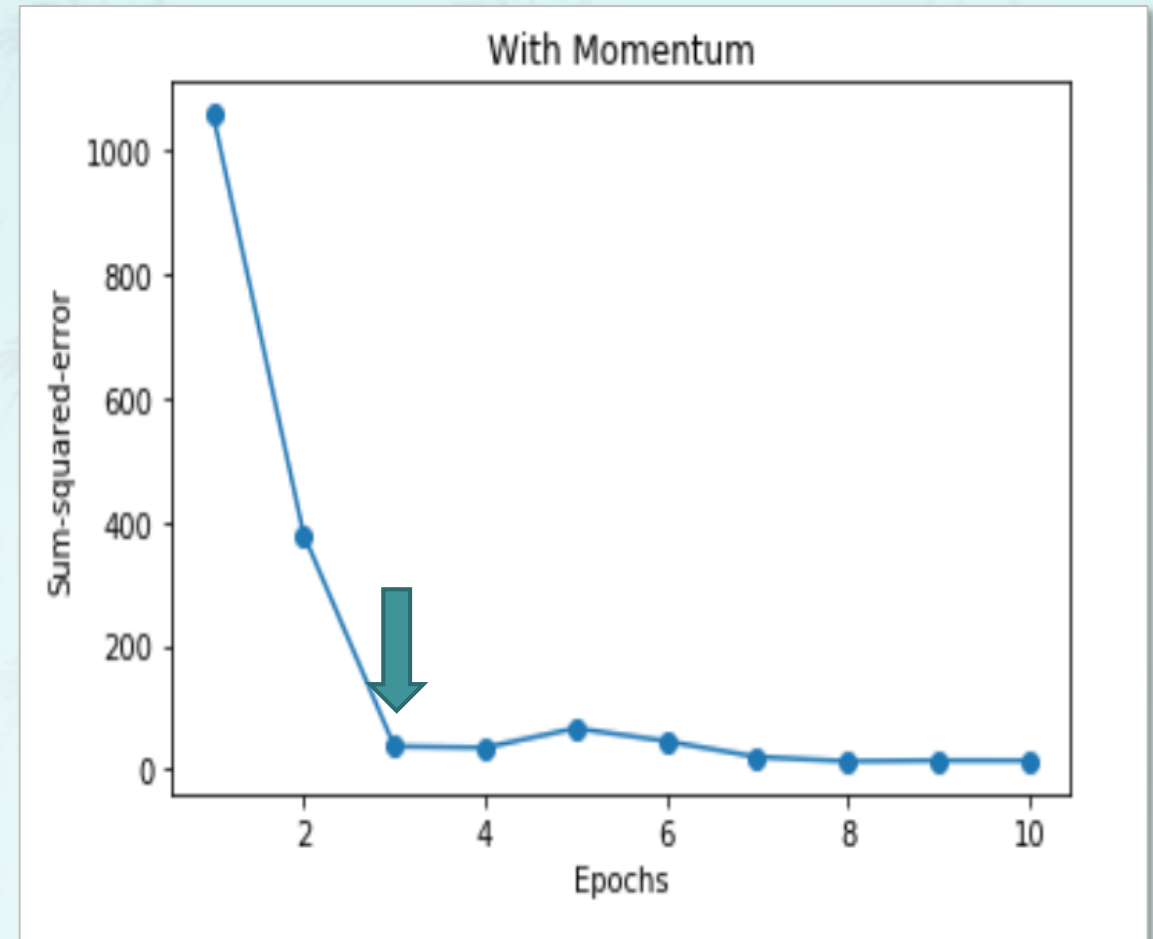


4. 모멘텀 : 적용 예시 (1)

모멘텀 적용 전

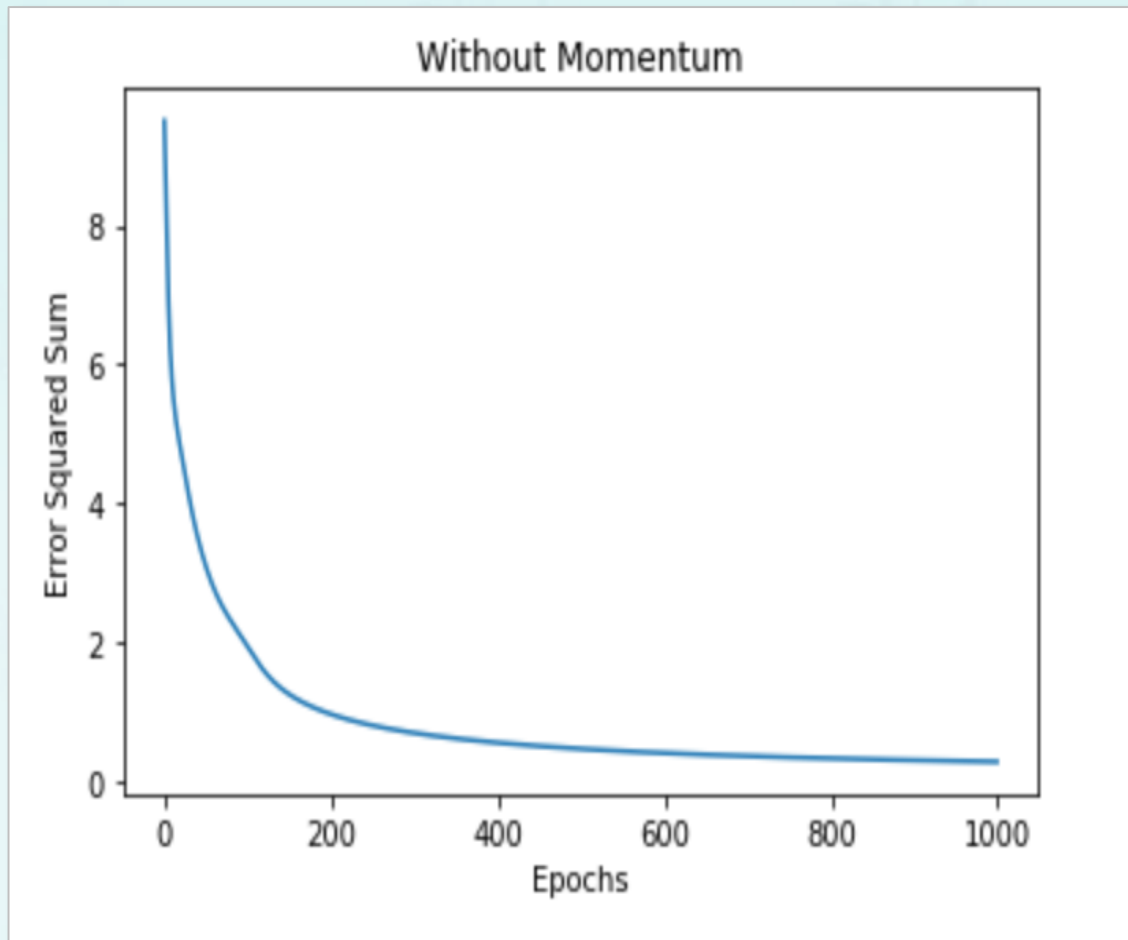


모멘텀 적용 후 (**gamma = 0.5**)

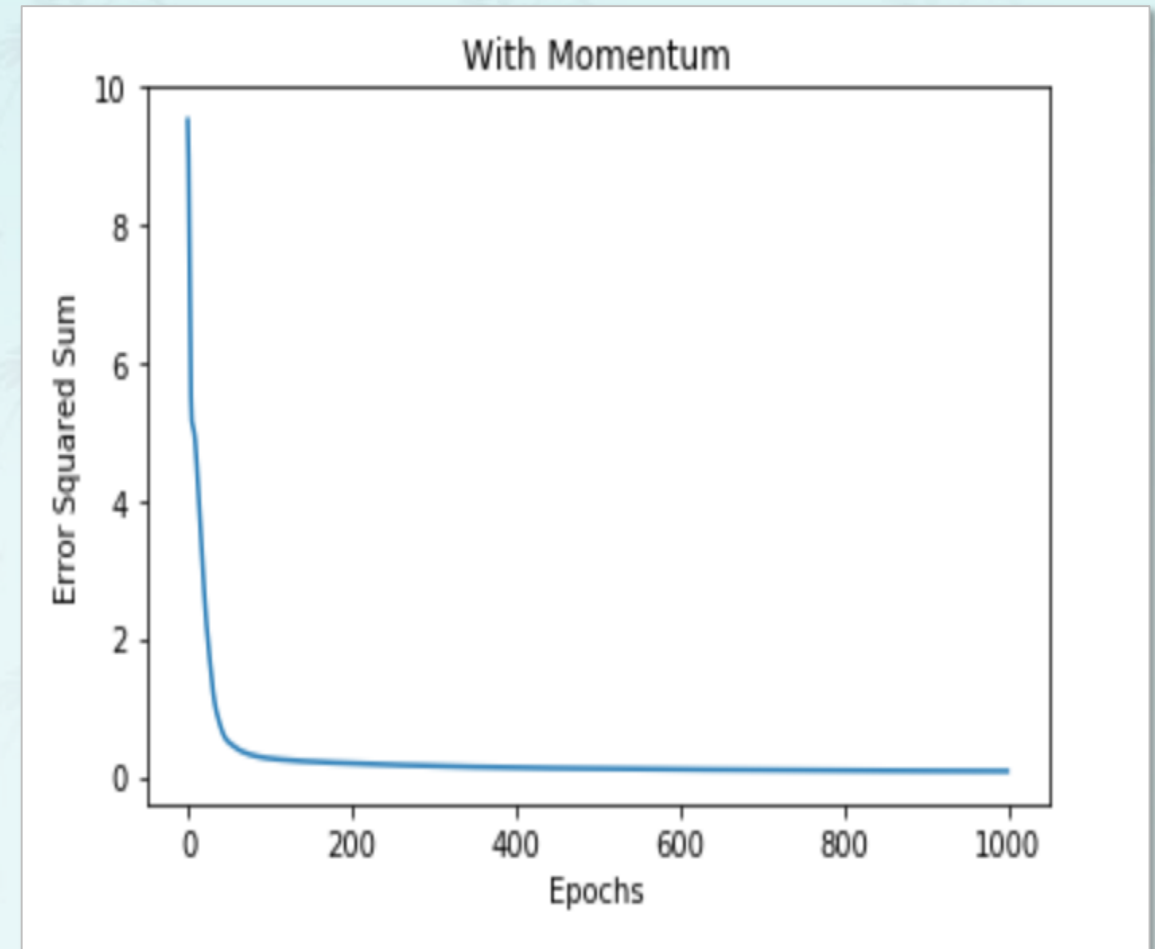


4. 모멘텀 : 적용 예시 (2)

모멘텀 적용 전

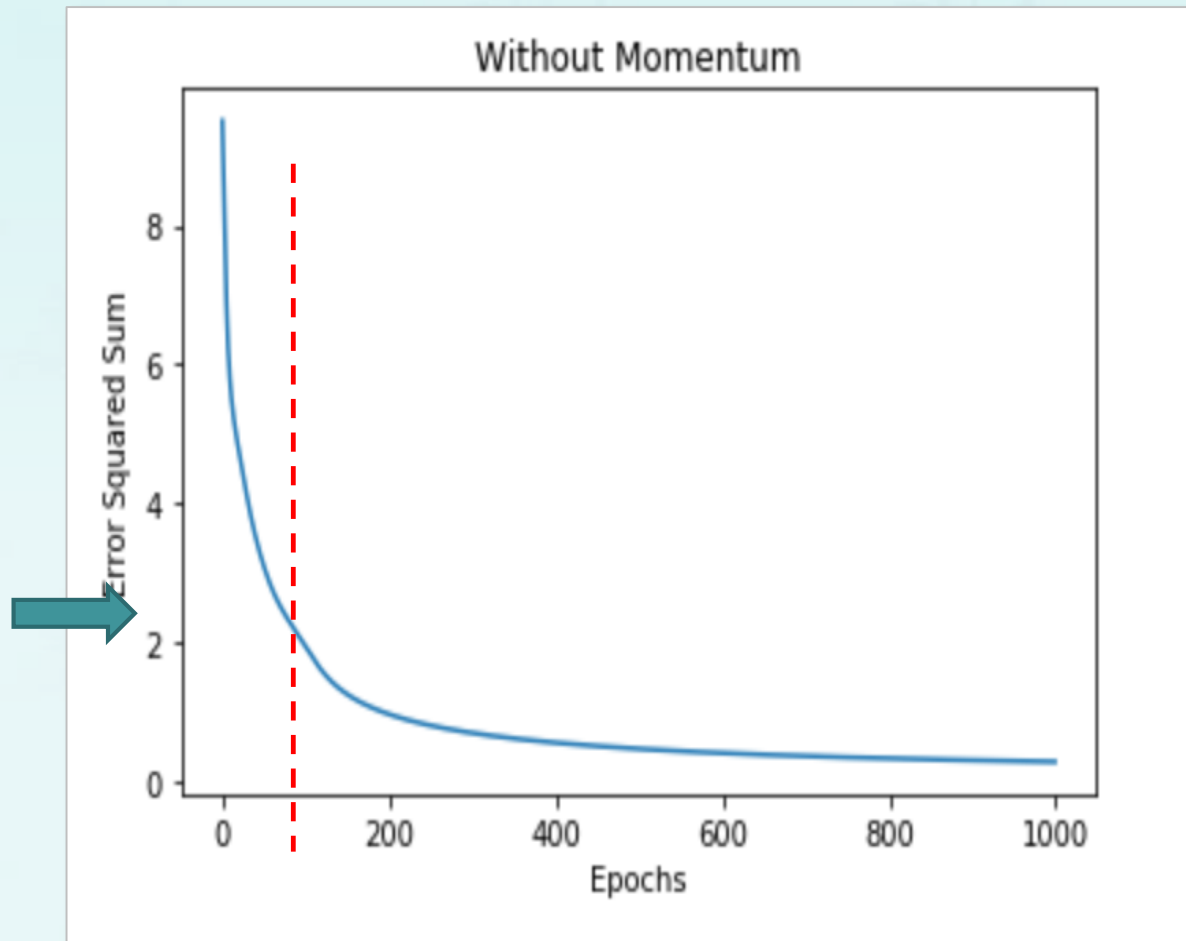


모멘텀 적용 후

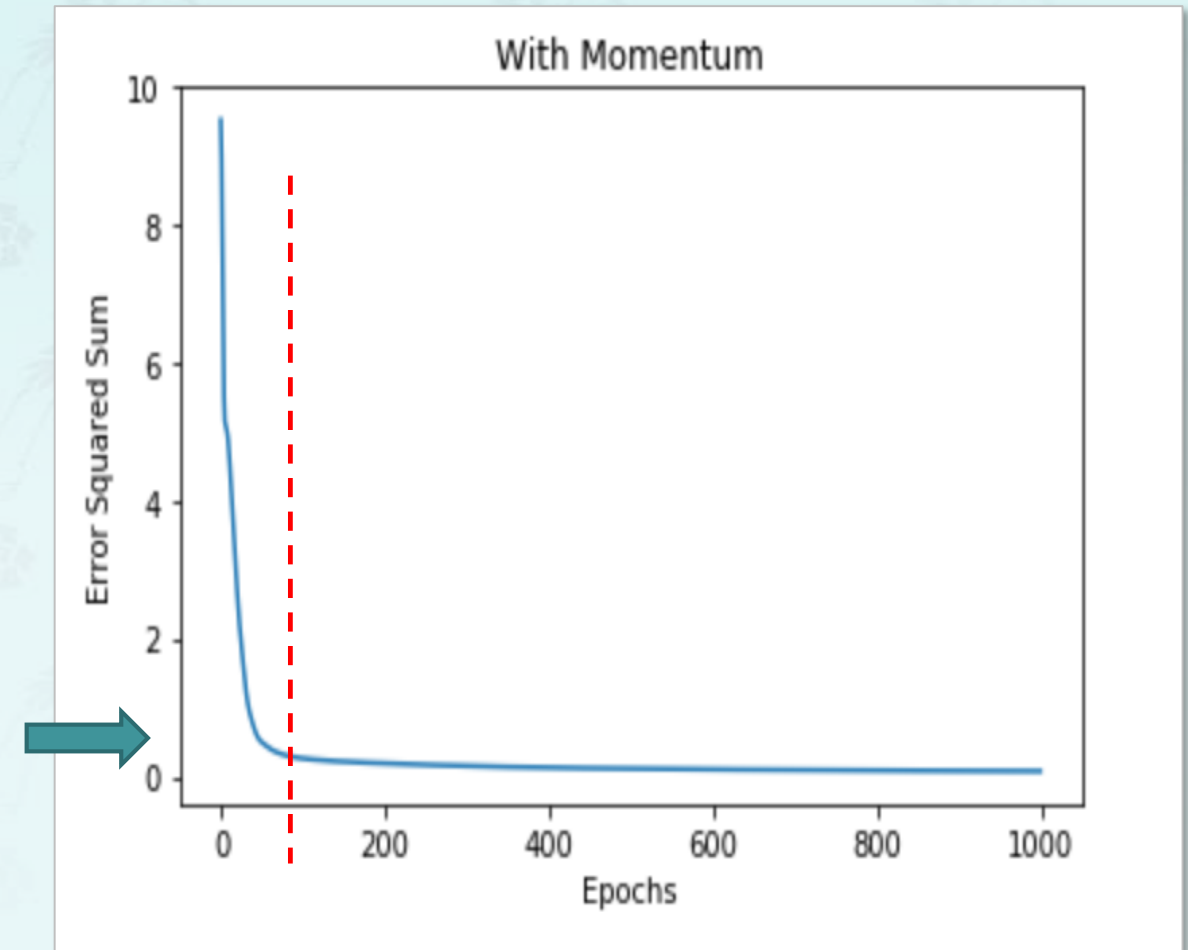


4. 모멘텀 : 적용 예시 (3)

모멘텀 적용 전



모멘텀 적용 후



4. 모멘텀: 수식표현

- $v = \gamma v + \eta \frac{dJ(w)}{dw}$
- v : 속력
- γ : 가속도

4. 모멘텀: 코드 변환

■ 모멘텀

- $v = \gamma v + \eta \frac{dJ(w)}{dw}$
- v : 속력
- γ : 가속도



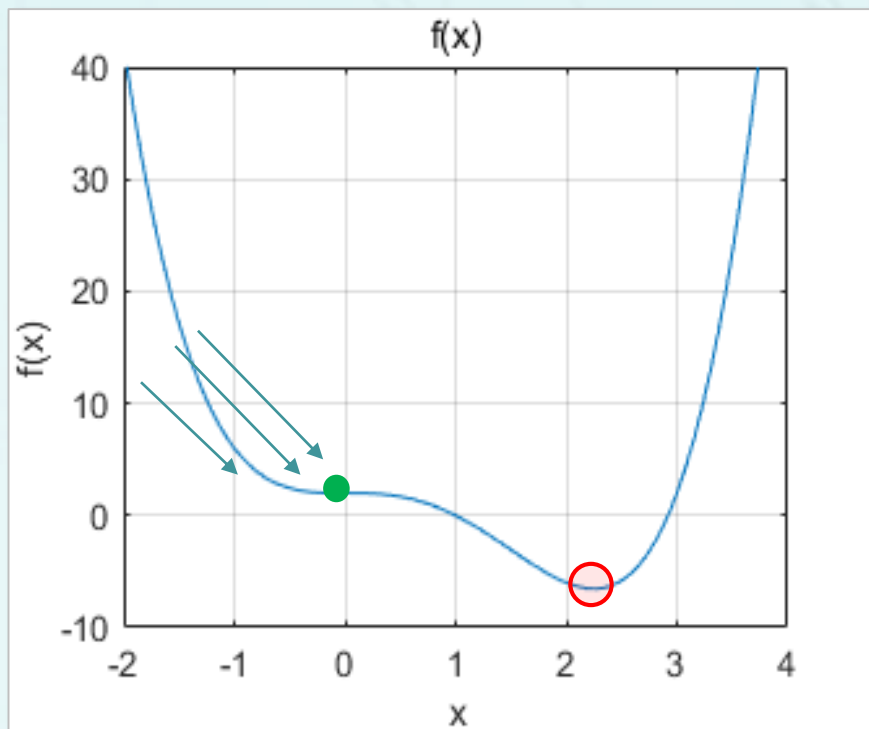
```
1 def fit(self, X, y):
2     np.random.RandomState(self.random_seed)
3     self.w = np.random.random(size=X.shape[1] + 1)
4
5     self.maxy, self.miny = y.max(), y.min()
6     self.cost_ = []
7     self.w_ = np.array([self.w])
8
9     """Momentum"""
10    self.v1 = np.zeros_like(self.w[1:])
11    self.v2 = np.zeros_like(self.w[0])
12    gamma = 0.5
13
14    for i in range(self.epochs):
15        yhat = self.activation(self.net_input(X))
16        errors = (y - yhat)
17
18        self.v1 = gamma*self.v1 + self.eta*np.dot(errors, X)
19        self.v2 = gamma*self.v2 + self.eta*np.sum(errors)
20
21        self.w[1:] += self.v1
22        self.w[0] += self.v2
23        cost = 0.5 * np.sum(errors**2)
24        self.cost_.append(cost)
25        self.w_ = np.vstack([self.w_, self.w])
26    return self
```

6. 모멘텀: fit 4. 모멘텀: 코드 변환

■ 모멘텀

$$v = \gamma v + \eta \frac{dJ(w)}{dw}$$

- v : 속력
- γ : 가속도



```
1 def fit(self, X, y):
2     np.random.RandomState(self.random_seed)
3     self.w = np.random.random(size=X.shape[1] + 1)
4
5     self.maxy, self.miny = y.max(), y.min()
6     self.cost_ = []
7     self.w_ = np.array([self.w])
8
9     """Momentum"""
10    self.v1 = np.zeros_like(self.w[1:])
11    self.v2 = np.zeros_like(self.w[0])
12    gamma = 0.5
13
14    for i in range(self.epochs):
15        yhat = self.activation(self.net_input(X))
16        errors = (y - yhat)
17
18        self.v1 = gamma*self.v1 + self.eta*np.dot(errors, X)
19        self.v2 = gamma*self.v2 + self.eta*np.sum(errors)
20
21        self.w[1:] += self.v1
22        self.w[0] += self.v2
23        cost = 0.5 * np.sum(errors**2)
24        self.cost_.append(cost)
25        self.w_ = np.vstack([self.w_, self.w])
26    return self
```

6. 모멘텀: fit 함수

```
1 def fit(self, X, y):
2     np.random.RandomState(self.random_seed)
3     self.w = np.random.random(size=X.shape[1] + 1)
4
5     self.maxy, self.miny = y.max(), y.min()
6     self.cost_ = []
7     self.w_ = np.array([self.w])
8
9     """Momentum"""
10    self.v1 = np.zeros_like(self.w[1:])
11    self.v2 = np.zeros_like(self.w[0])
12    gamma = 0.5
13
14    for i in range(self.epochs):
15        yhat = self.activation(self.net_input(X))
16        errors = (y - yhat)
17
18        self.v1 = gamma*self.v1 + self.eta*np.dot(errors, X)
19        self.v2 = gamma*self.v2 + self.eta*np.sum(errors)
20
21        self.w[1:] += self.v1
22        self.w[0] += self.v2
23        cost = 0.5 * np.sum(errors**2)
24        self.cost_.append(cost)
25        self.w_ = np.vstack([self.w_, self.w])
26    return self
```


8-2 아달라인 경사하강법의 적용

- 학습 정리
 - 붓꽃 학습자료 속성
 - 붓꽃 학습자료 예제
 - 아달라인 객체 생성과 테스트
 - 지역 최소와 전역 최소
 - 모멘텀