9주차(2/3)

아달라인 경사하강법 구현

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

1. 아달라인 경사하강법의 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. 아달라인 경사하강법의 적용: 붓꽃 학습자료

- 붓꽃 학습자료

(The use of multiple measurements in taxonomic problems, Ronald Fisher)

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료

- 붓꽃(세토사, 버시칼라, 버지니카)

(The use of multiple measurements in taxonomic problems, Ronald Fisher)

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료

- 붓꽃(세토사, 버시칼라, 버지니카)

- 특성
  - 꽃잎의 길이, 너비
  - 꽃받침의 길이, 너비
  - 붓꽃 종류의 이름

(The use of multiple measurements in taxonomic problems, Ronald Fisher)

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료

- 특성행렬
  - 행 : 샘플 수
  - 열 : 특성 수

$$\mathrm{X} \in \mathbb{R}^{150\times4}$$

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

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료

- 특성행렬
  - 행 : 샘플 수
  - 열 : 특성 수

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

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

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료

- 특성행렬
  - 행 : 샘플 수
  - 열 : 특성 수
  - 형상(샘플의 수 x 특성의 수)

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

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

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

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

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료

- 특성행렬
  - 행 : 샘플 수
  - 열 : 특성 수

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

$$y \in big(Setosa, Vericolor, Virginica)$$

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료

- 붓꽃 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. 아달라인 경사하강법의 적용: 붓꽃 학습자료

- 붓꽃 자료 읽기

```python
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

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료

- 붓꽃 자료 읽기

```python
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

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료

- 붓꽃 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

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료 예제

- 아달라인 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)
```

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료 예제

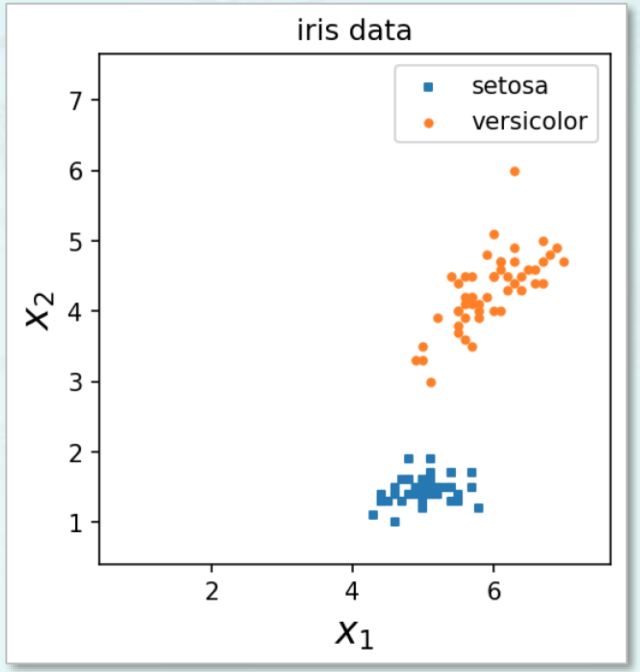
- 아달라인 학습

```
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)
```

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료 예제

- 아달라인 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)
```



1. 아달라인 경사하강법의 적용: 붓꽃 학습자료 예제

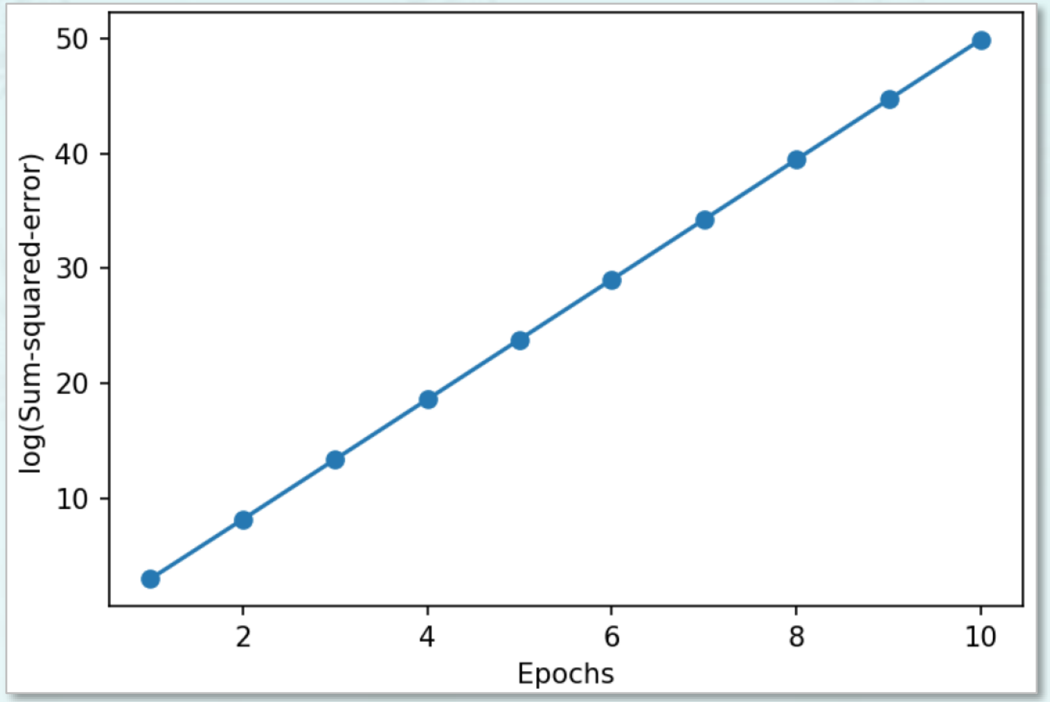
- 아달라인 학습

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

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료 예제

- 아달라인 학습

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



1. 아달라인 경사하강법의 적용: 붓꽃 학습자료 예제

- 학습률 (η : ↑)

```
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()
```

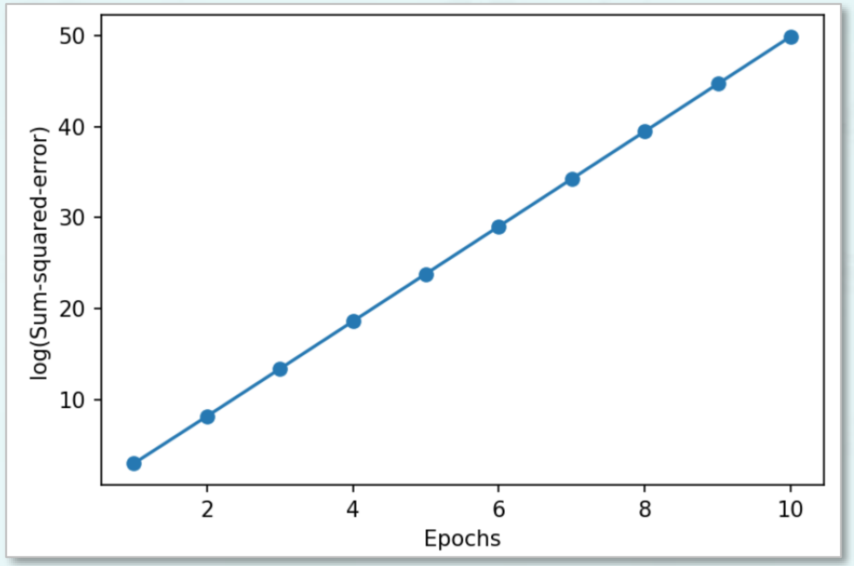
- 학습률 (η : ↓)

```
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()
```

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료 예제

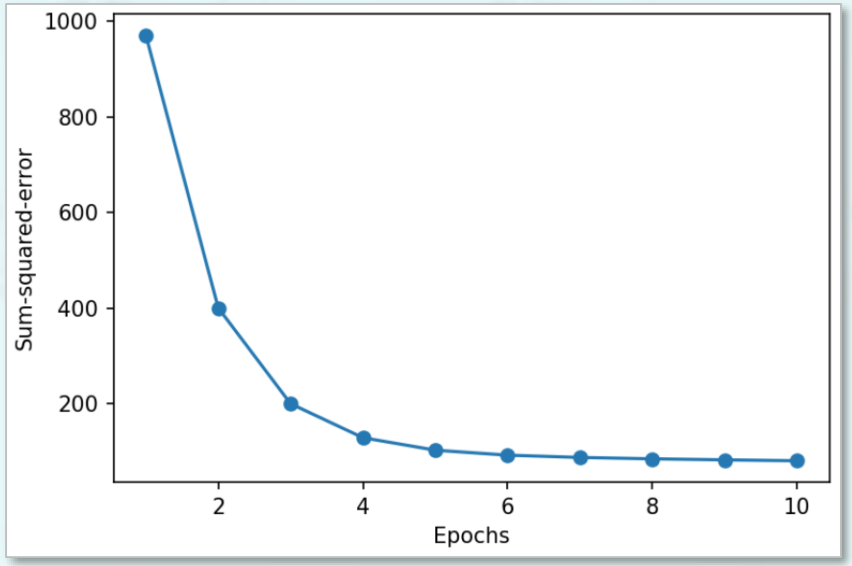
- 학습률 (η : ↑)

```
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 (η : ↓)

```
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()
```



1. 아달라인 경사하강법의 적용: 붓꽃 학습자료 예제

- 붓꽃자료의 전처리 (표준화)

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료 예제

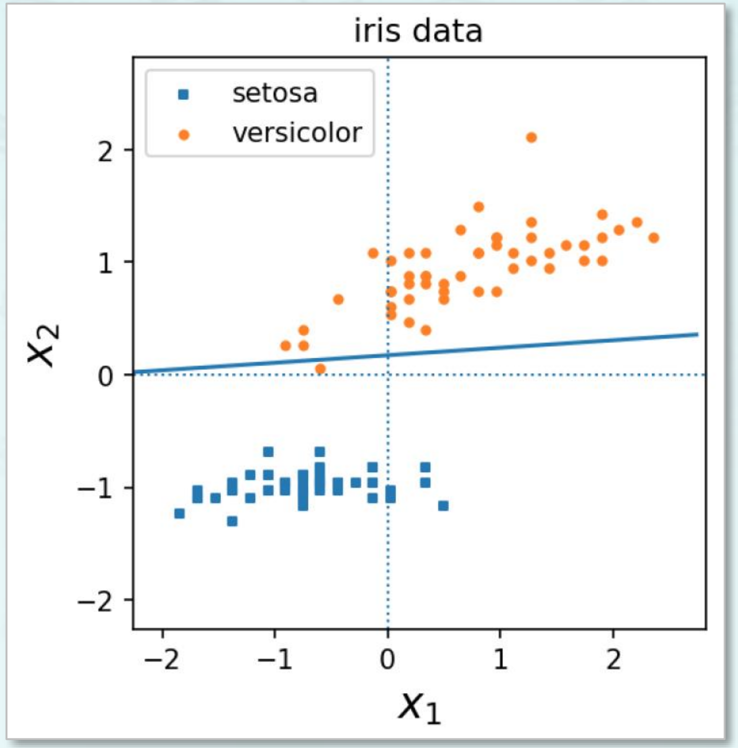
- 붓꽃자료의 전처리 (표준화)

```python
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)
```

1. 아달라인 경사하강법의 적용: 붓꽃 학습자료 예제

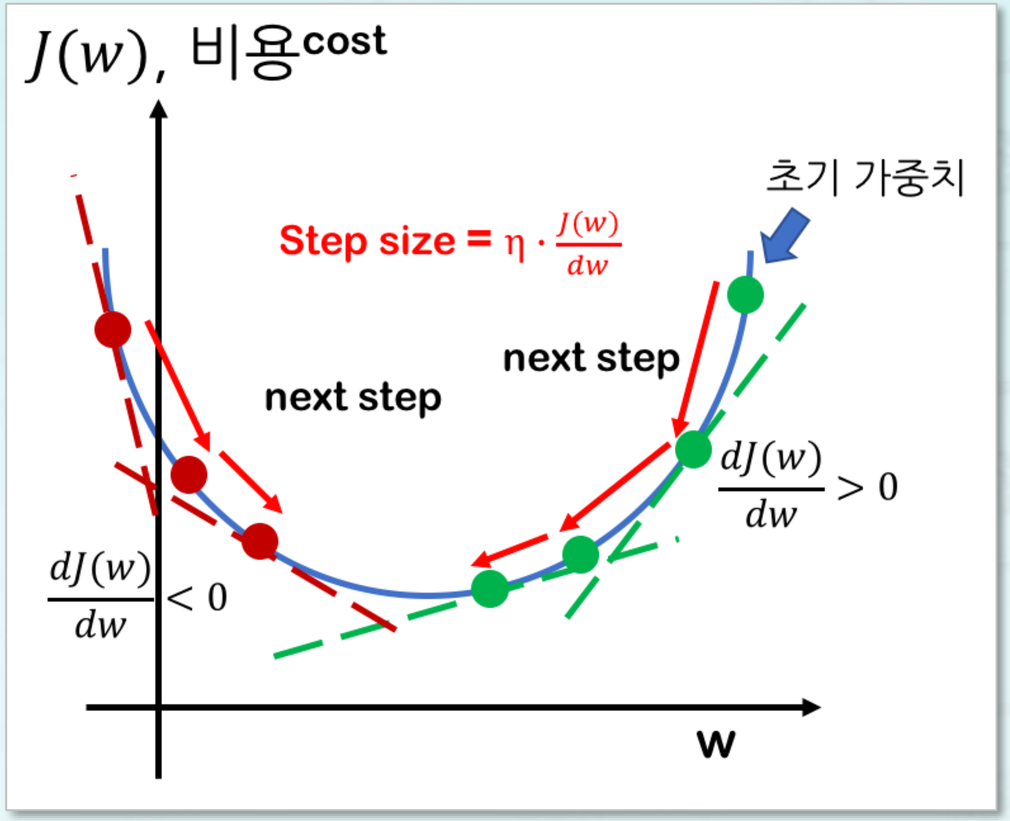
- 붓꽃자료의 전처리 (표준화)

```
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)
```



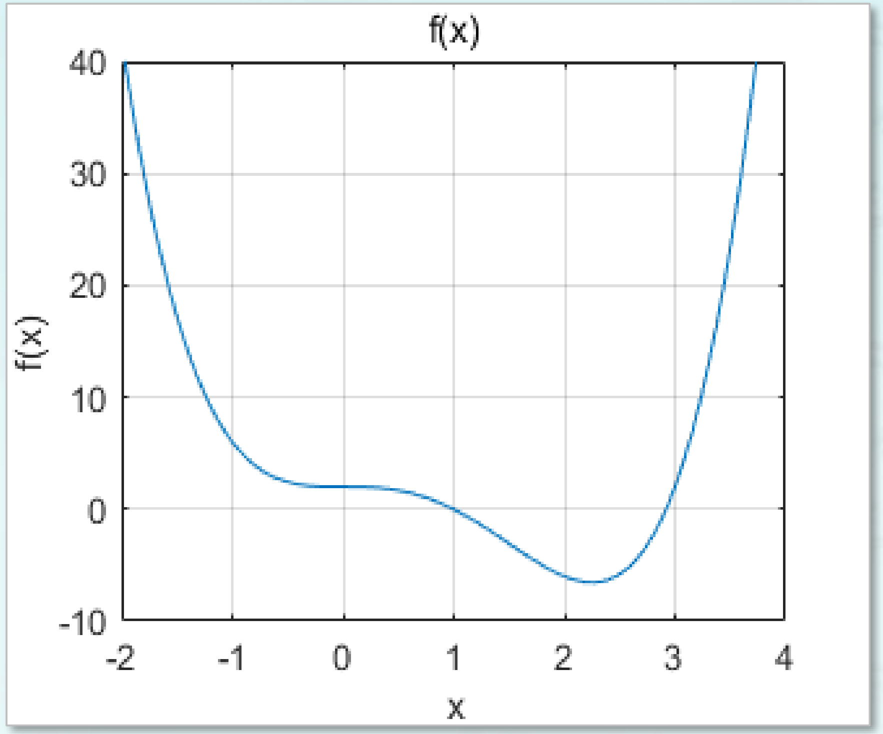
2. 모멘텀: 지역 최소와 전역 최소

- 경사하강법



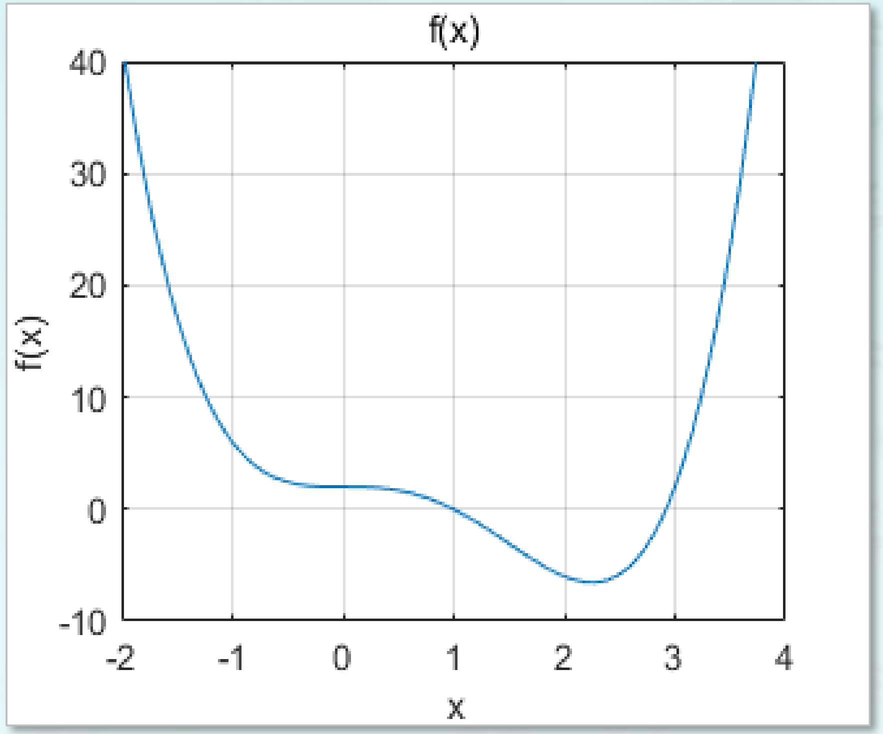
2. 모멘텀: 지역 최소와 전역 최소

- 경사하강법
  - $f(x) = x^4 - 3x^3 + 2$

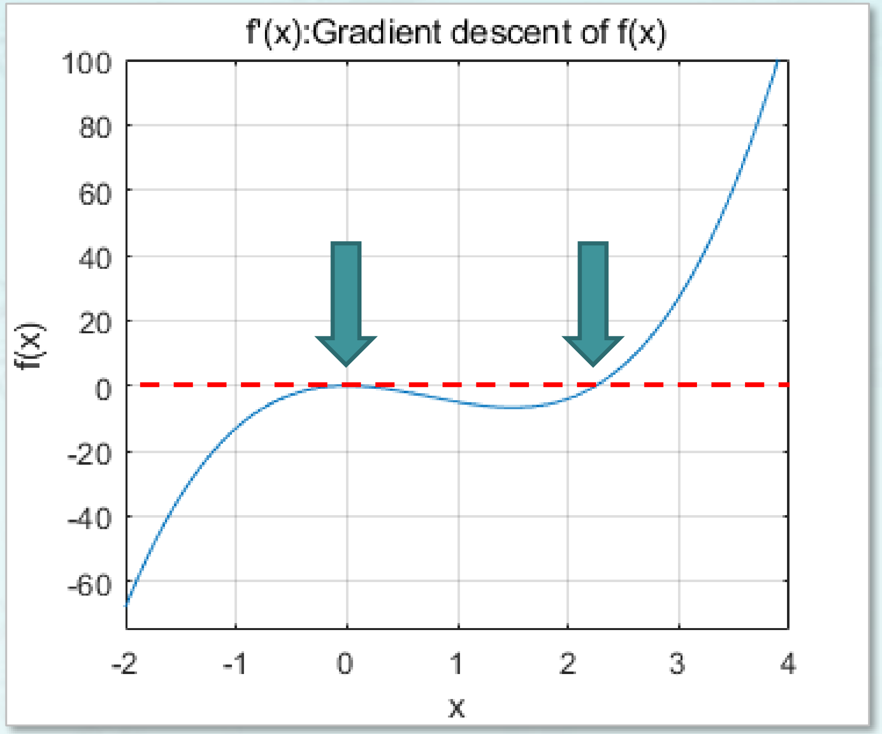


2. 모멘텀: 지역 최소와 전역 최소

- 경사하강법
  - $f(x) = x^4 - 3x^3 + 2$

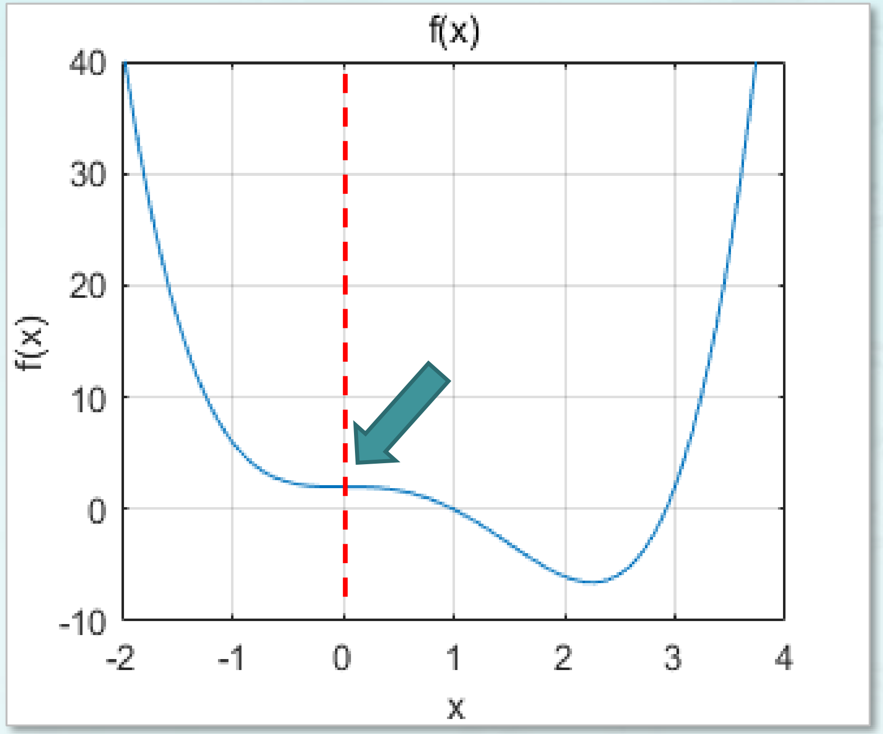


- 기울기
  - $f'(x) = 4x^3 - 9x^2$

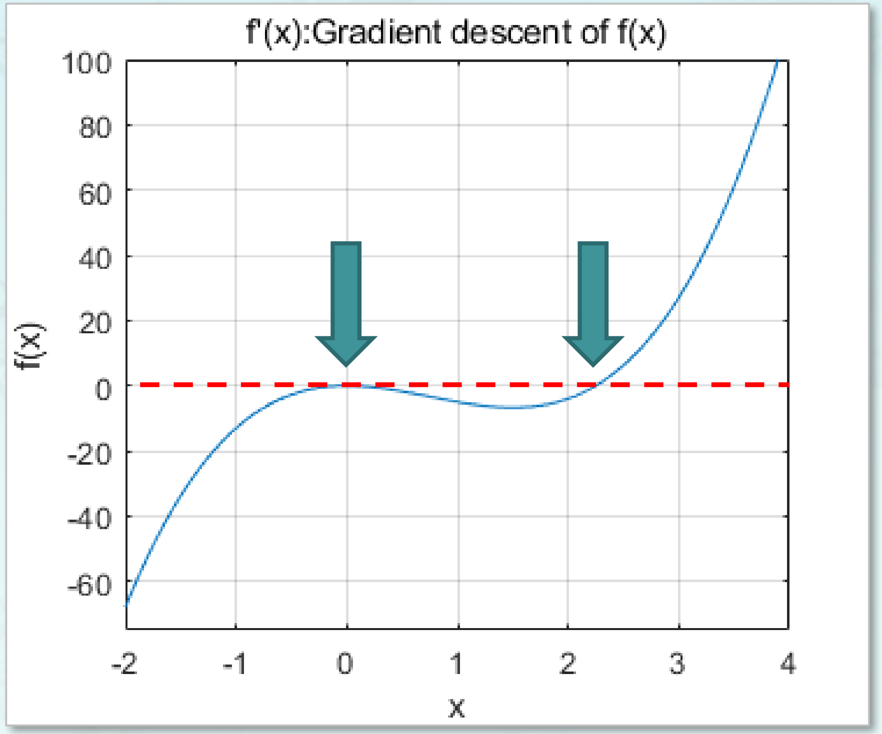


2. 모멘텀: 지역 최소와 전역 최소

- 경사하강법
  - $f(x) = x^4 - 3x^3 + 2$

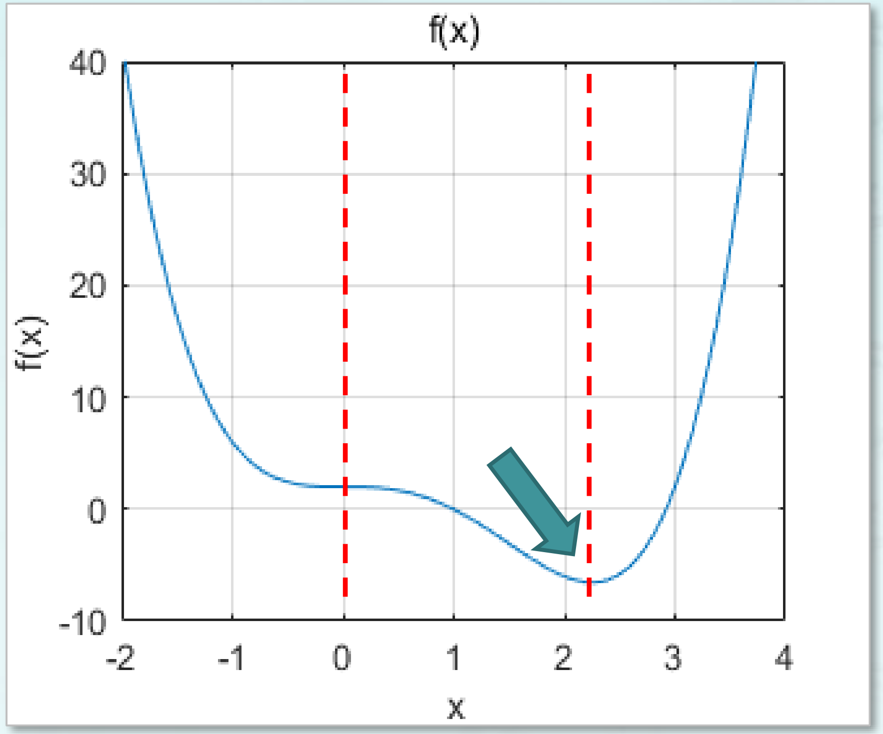


- 기울기
  - $f'(x) = 4x^3 - 9x^2$

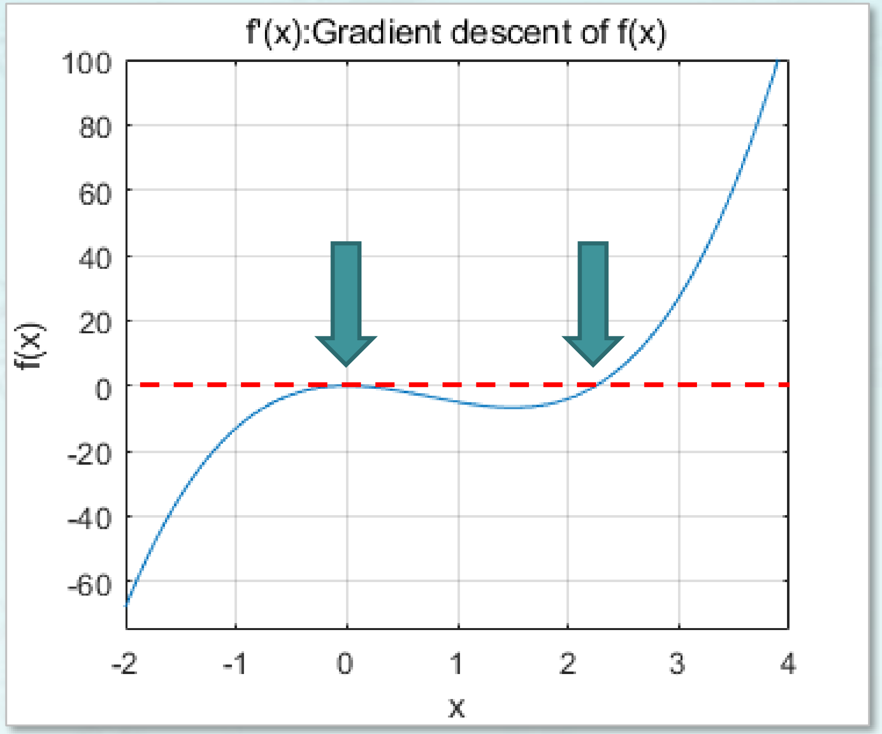


2. 모멘텀: 지역 최소와 전역 최소

- 경사하강법
  - $f(x) = x^4 - 3x^3 + 2$

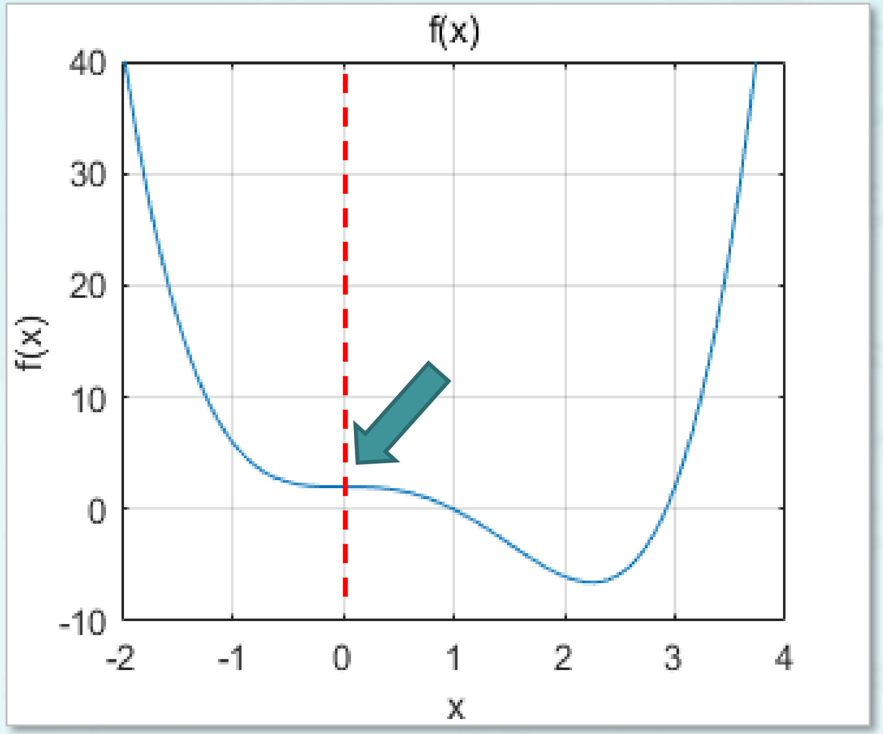


- 기울기
  - $f'(x) = 4x^3 - 9x^2$



2. 모멘텀: 지역 최소와 전역 최소

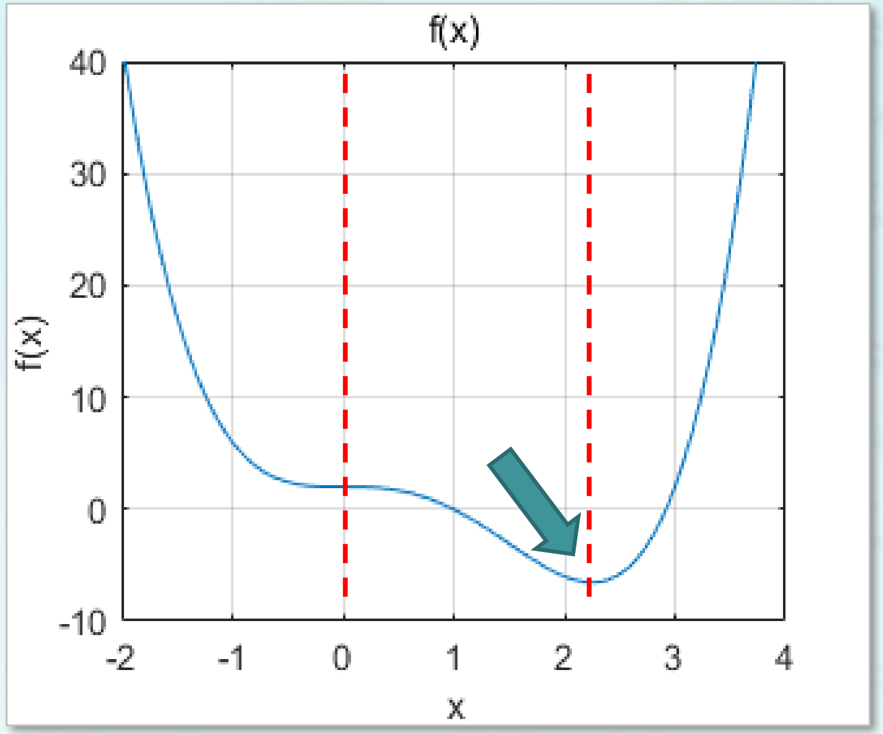
- 경사하강법
  - $f(x) = x^4 - 3x^3 + 2$



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

2. 모멘텀: 지역 최소와 전역 최소

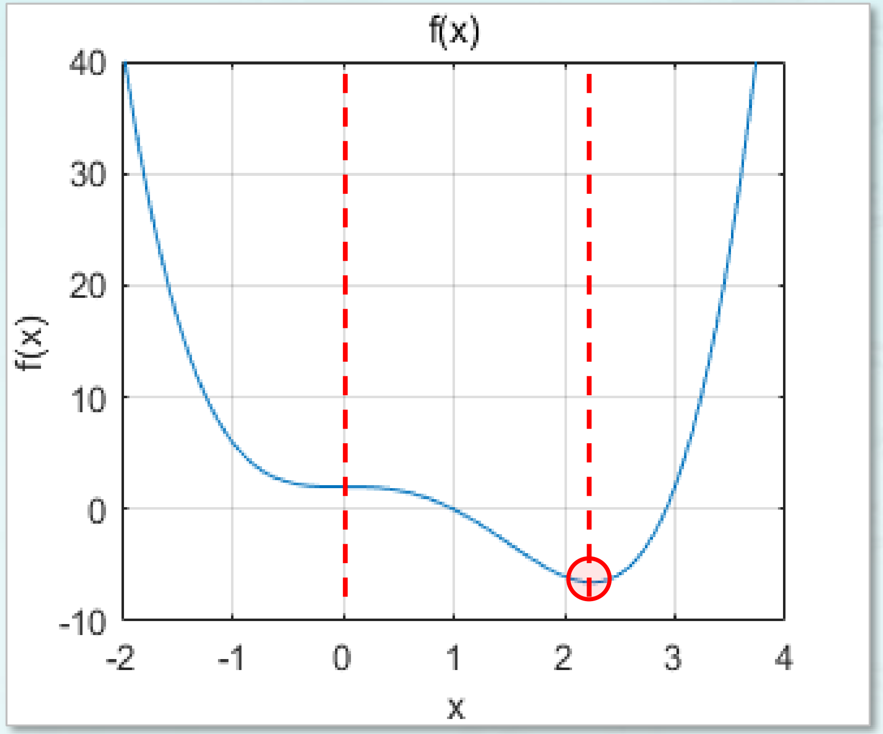
- 경사하강법
  - $f(x) = x^4 - 3x^3 + 2$



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

2. 모멘텀: 지역 최소와 전역 최소

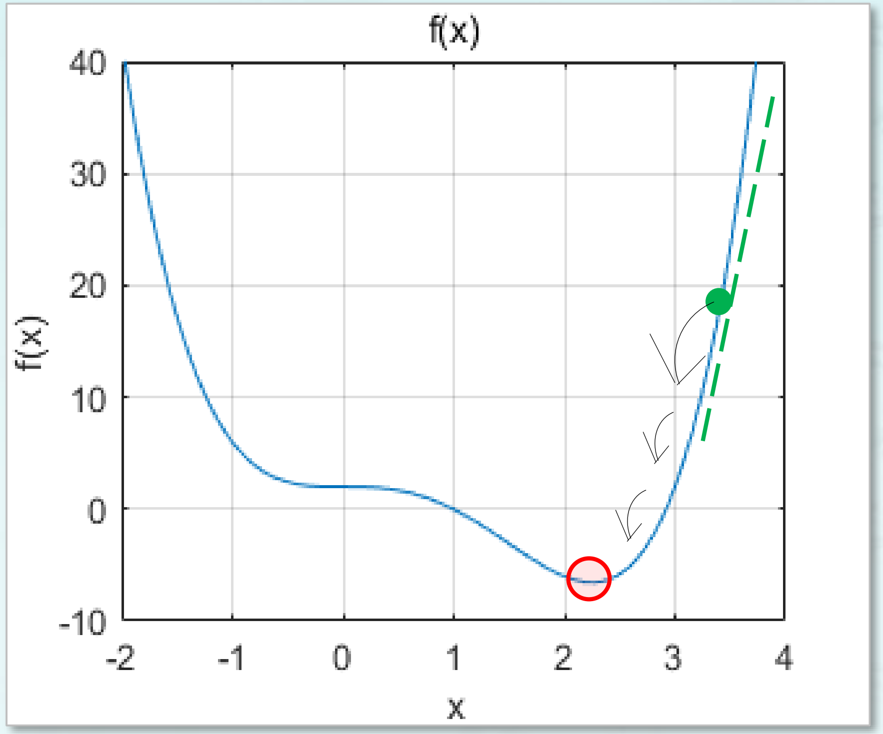
- 경사하강법
  - $f(x) = x^4 - 3x^3 + 2$



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

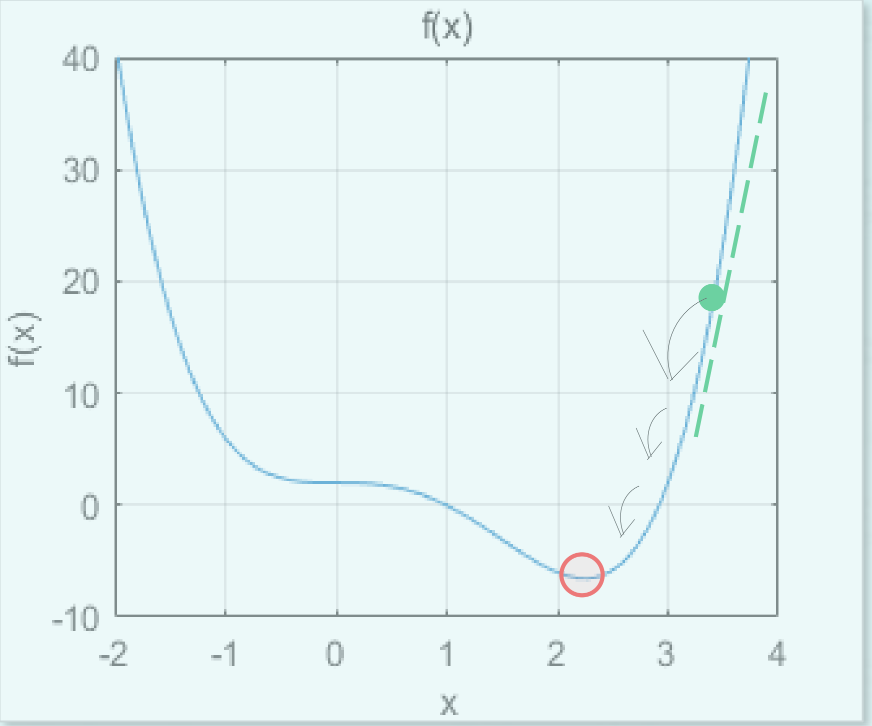
2. 모멘텀: 지역 최소와 전역 최소

- 전역 최소(**Global Minimum)**

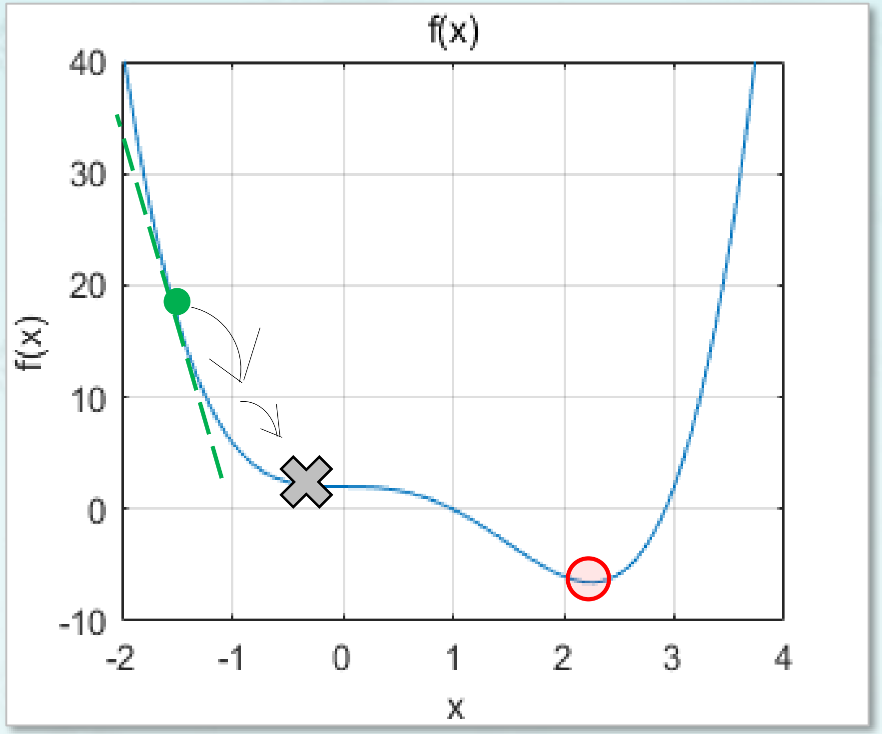


2. 모멘텀: 지역 최소와 전역 최소

- 전역 최소(Global Minimum)

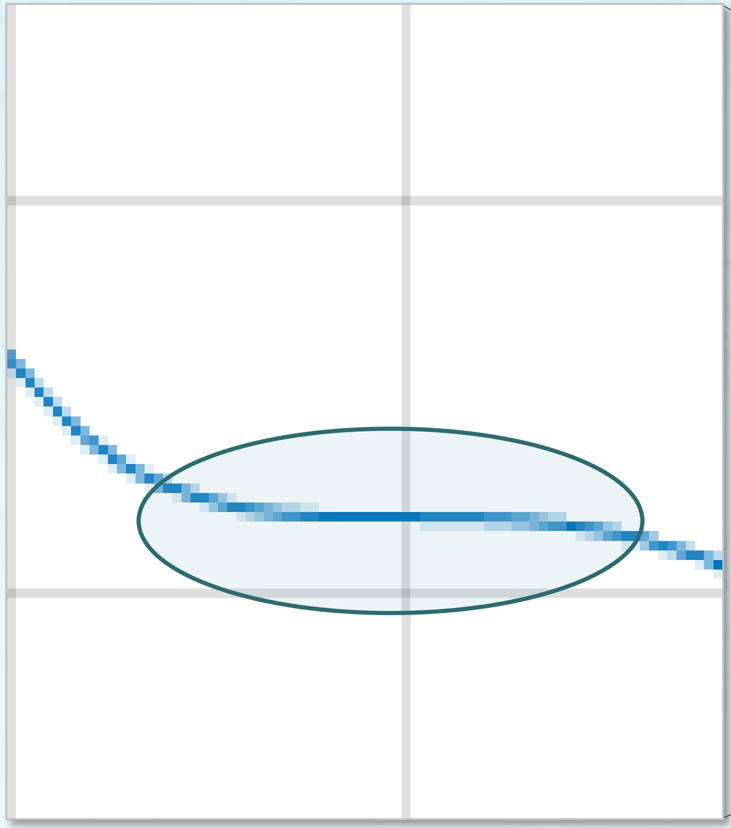


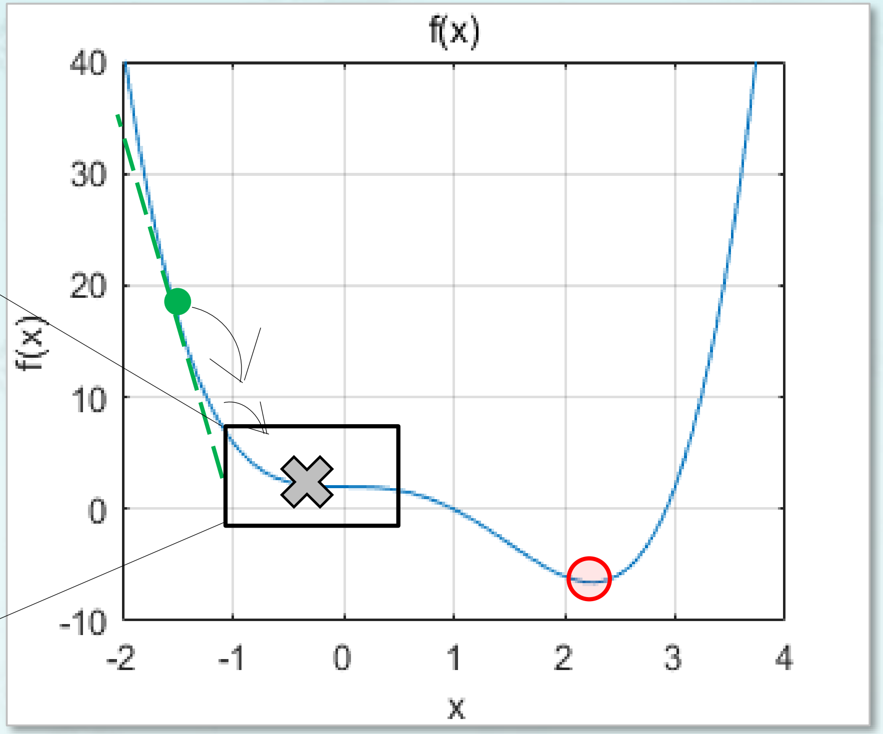
- 지역 최소(Local Minimum)



2. 모멘텀: 지역 최소와 전역 최소

- 지역 최소**(Local Minimum)**



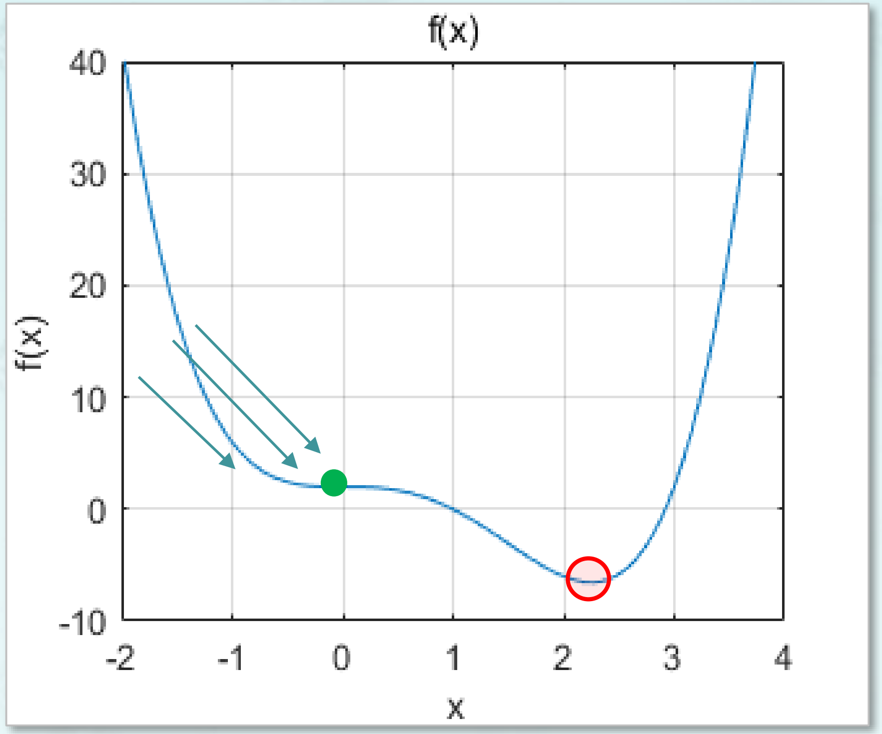


2. 모멘텀

- 모멘텀**(Momentum)**

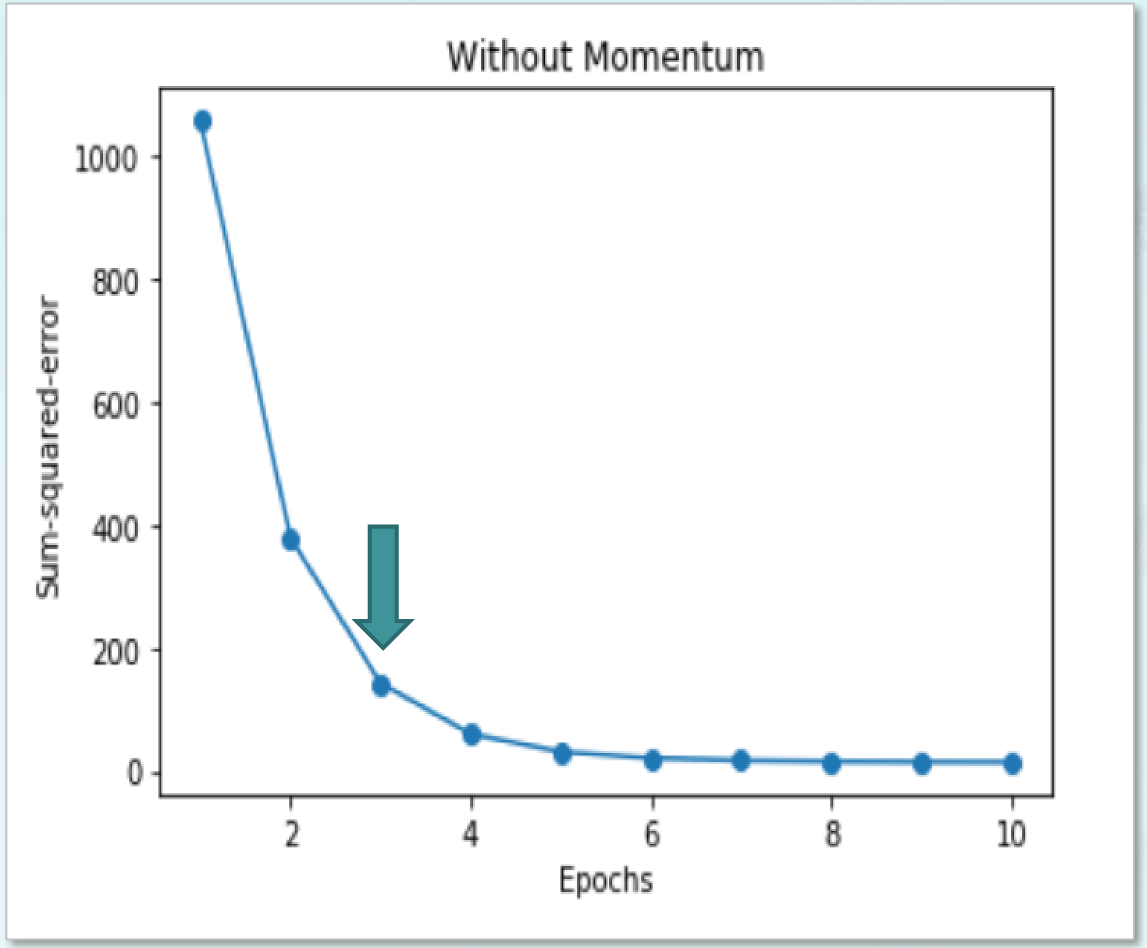
2. 모멘텀

- 모멘텀(**Momentum**)

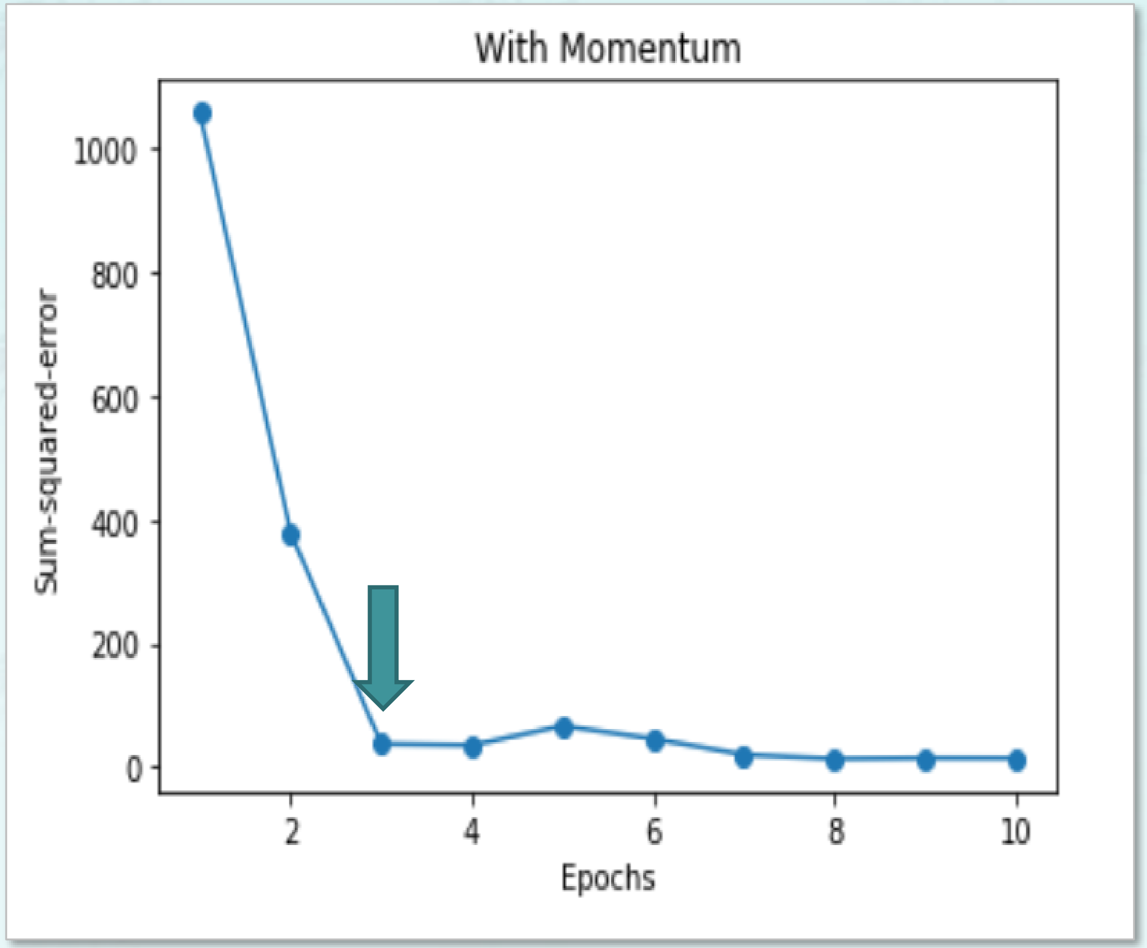


2. 모멘텀

- 0.5



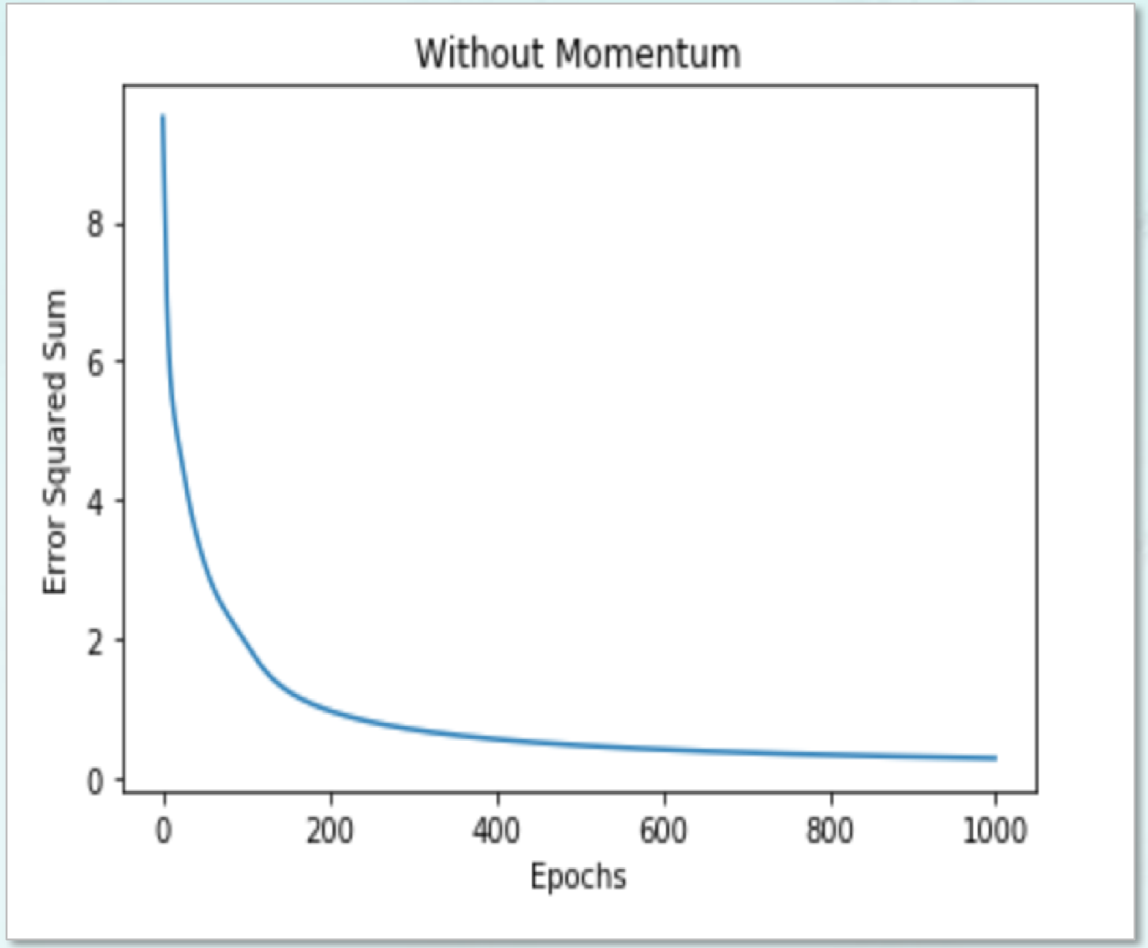
Without Momentum
Sum-squared-error
0
200
400
600
800
1000
2
4
6
8
10
Epochs



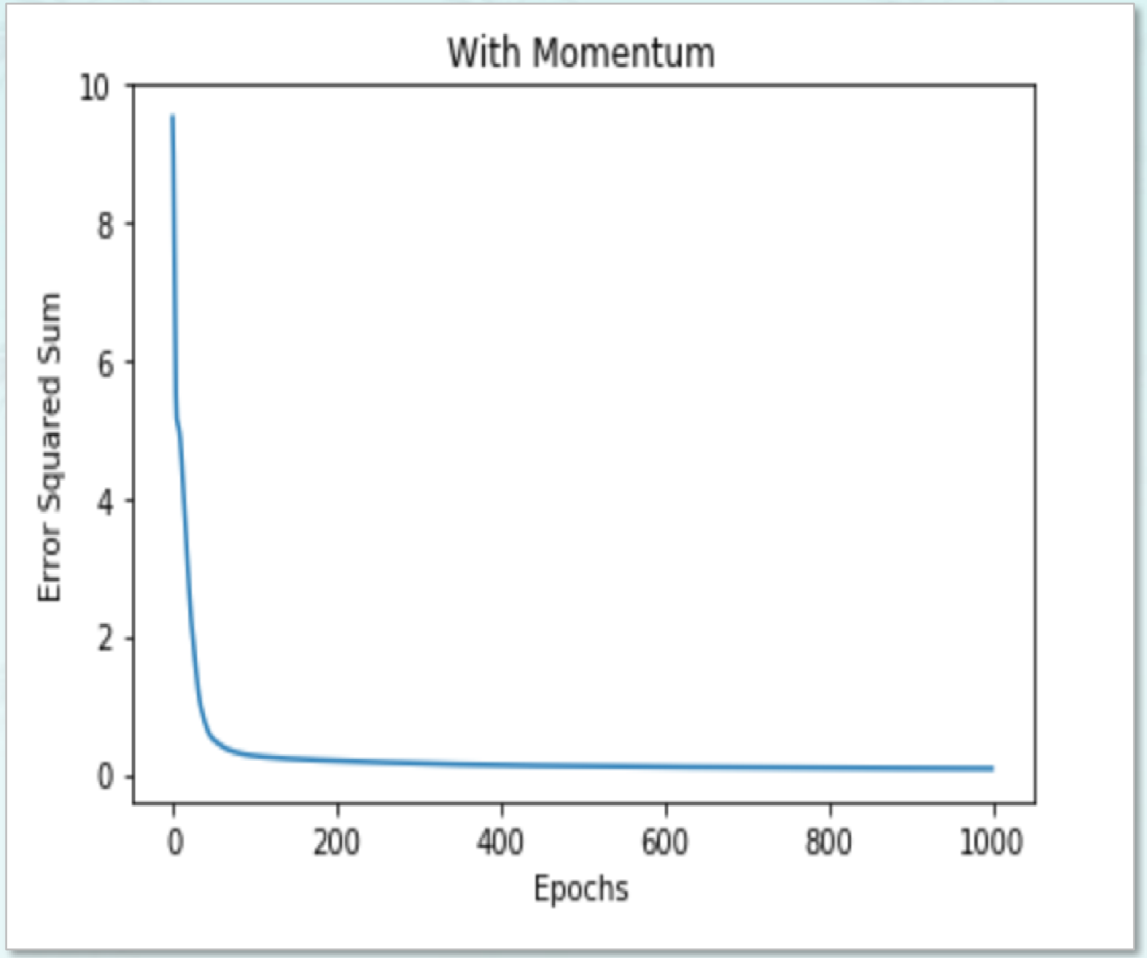
With Momentum
Sum-squared-error
0
200
400
600
800
1000
2
4
6
8
10
Epochs

2. 모멘텀

- 모멘텀 없는 경사하강법

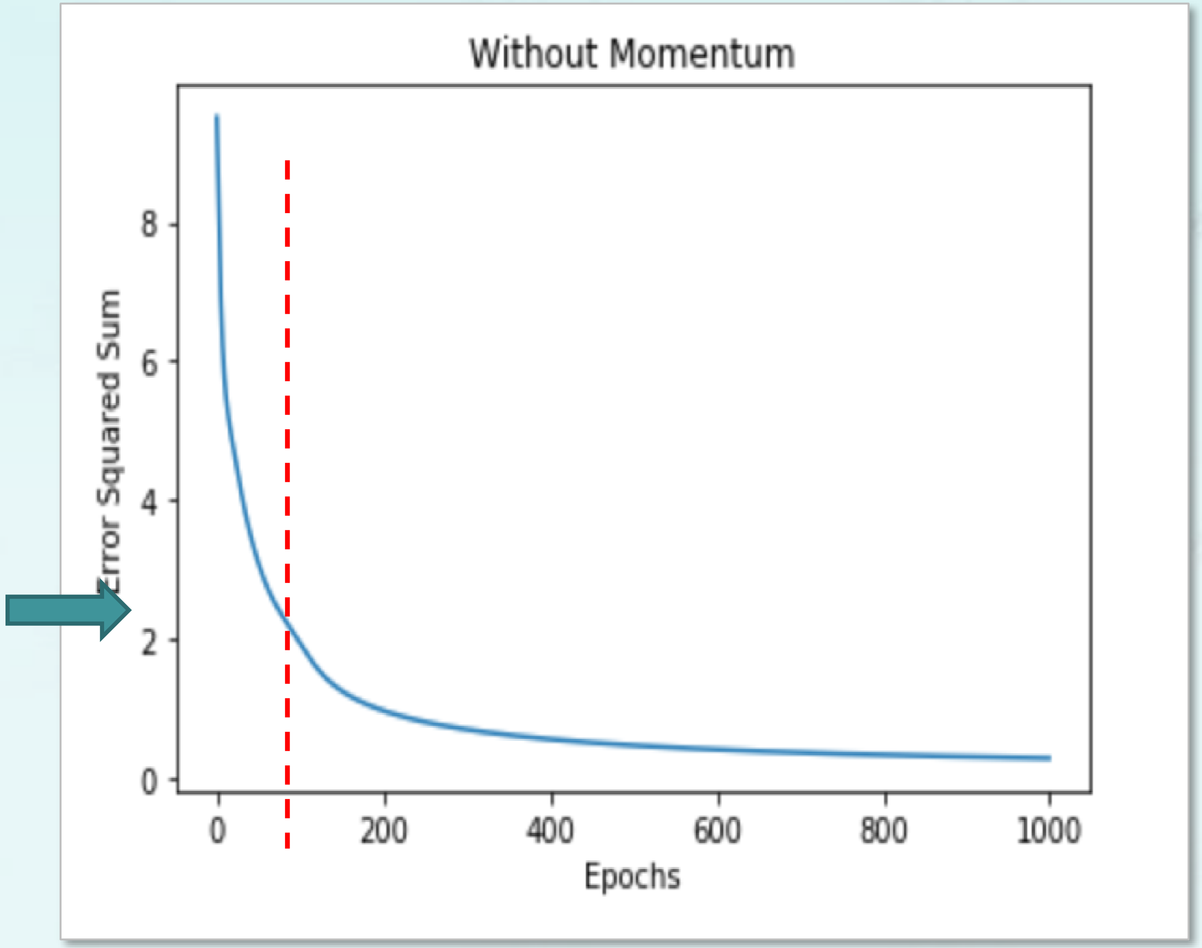


- 모멘텀을 이용한 경사하강법

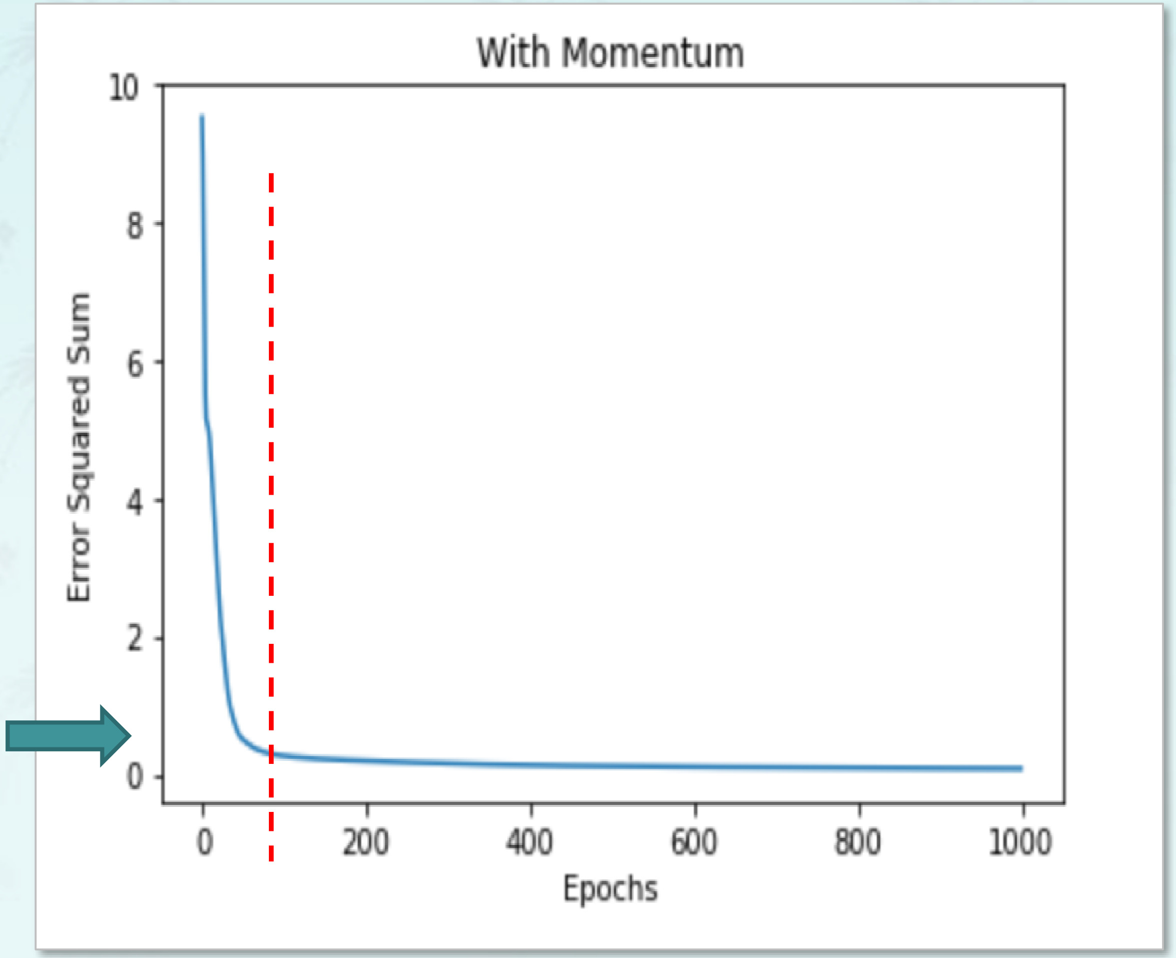


2. 모멘텀

- 모멘텀 없는 경사하강법



- 모멘텀을 이용한 경사하강법



2. 모멘텀: 정의와 구현

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

2. 모멘텀: 정의와 구현

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

```
def fit(self, X, y, X0=False):
    if X0 == False:
        X = np.c_[np.ones(len(y)), X]

    np.random.seed(self.random_seed)
    self.w = np.random.random(X.shape[1])

    self.maxy, self.miny = y.max(), y.min()
    self.cost_ = []
    self.w_ = np.array([self.w])

    """Momentum"""
    self.v1 = np.zeros_like(self.w[1:])
    self.v2 = np.zeros_like(self.w[0])
    gamma = 0.5

    for i in range(self.epochs):
        yhat = self.activation(self.net_input(X))
        errors = (y - yhat)

        self.v1 = gamma * self.v1 + self.eta * np.dot(errors, X)
        self.v2 = gamma * self.v2 + self.eta * np.sum(errors)

        self.w[1:] += self.v1
        self.w[0] += self.v2
        cost = 0.5 * np.sum(errors**2)
        self.cost_.append(cost)
        self.w_ = np.vstack([self.w_, self.w])
    return self
```

2. 모멘텀: 정의와 구현

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

```python
def fit(self, X, y, X0=False):
    if X0 == False:
        X = np.c_[np.ones(len(y)), X]

    np.random.seed(self.random_seed)
    self.w = np.random.random(X.shape[1])

    self.maxy, self.miny = y.max(), y.min()
    self.cost_ = []
    self.w_ = np.array([self.w])

    """Momentum"""
    self.v1 = np.zeros_like(self.w[1:])
    self.v2 = np.zeros_like(self.w[0])
    gamma = 0.5

    for i in range(self.epochs):
        yhat = self.activation(self.net_input(X))
        errors = (y - yhat)

        self.v1 = gamma * self.v1 + self.eta * np.dot(errors, X)
        self.v2 = gamma * self.v2 + self.eta * np.sum(errors)

        self.w[1:] += self.v1
        self.w[0] += self.v2
        cost = 0.5 * np.sum(errors**2)
        self.cost_.append(cost)
        self.w_ = np.vstack([self.w_, self.w])
    return self
```

2. 모멘텀: 정의와 구현

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

```python
def fit(self, X, y, X0=False):
    if X0 == False:
        X = np.c_[np.ones(len(y)), X]

    np.random.seed(self.random_seed)
    self.w = np.random.random(X.shape[1])

    self.maxy, self.miny = y.max(), y.min()
    self.cost_ = []
    self.w_ = np.array([self.w])

    """Momentum"""
    self.v1 = np.zeros_like(self.w[1:])
    self.v2 = np.zeros_like(self.w[0])
    gamma = 0.5

    for i in range(self.epochs):
        yhat = self.activation(self.net_input(X))
        errors = (y - yhat)

        self.v1 = gamma * self.v1 + self.eta * np.dot(errors, X)
        self.v2 = gamma * self.v2 + self.eta * np.sum(errors)

        self.w[1:] += self.v1
        self.w[0] += self.v2
        cost = 0.5 * np.sum(errors**2)
        self.cost_.append(cost)
        self.w_ = np.vstack([self.w_, self.w])
    return self
```

2. 모멘텀: 정의와 구현

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

```
def fit(self, X, y, X0=False):
    if X0 == False:
        X = np.c_[np.ones(len(y)), X]

    np.random.seed(self.random_seed)
    self.w = np.random.random(X.shape[1])

    self.maxy, self.miny = y.max(), y.min()
    self.cost_ = []
    self.w_ = np.array([self.w])

    """Momentum"""
    self.v1 = np.zeros_like(self.w[1:])
    self.v2 = np.zeros_like(self.w[0])
    gamma = 0.5

    for i in range(self.epochs):
        yhat = self.activation(self.net_input(X))
        errors = (y - yhat)

        self.v1 = gamma * self.v1 + self.eta * np.dot(errors, X)
        self.v2 = gamma * self.v2 + self.eta * np.sum(errors)

        self.w[1:] += self.v1
        self.w[0] += self.v2
        cost = 0.5 * np.sum(errors**2)
        self.cost_.append(cost)
        self.w_ = np.vstack([self.w_, self.w])
    return self
```

아달라인 경사하강법의 적용

- 학습 정리
  - 붓꽃 학습자료를 이해하기
  - 아달라인 객체 생성과 테스트
  - 지역 최소, 전역 최소
  - 모멘텀

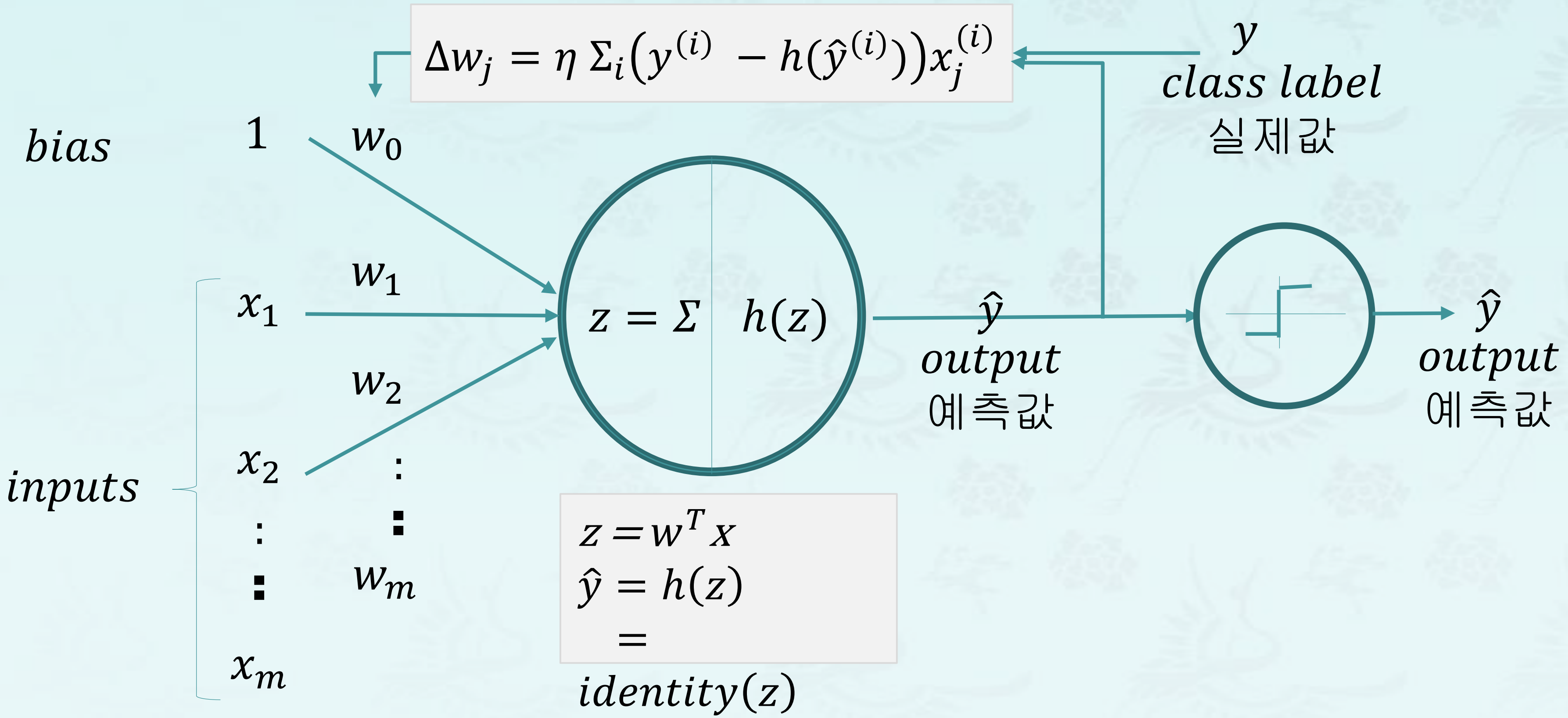
- **8-3 역전파(1)**

9주차(2/3)

아달라인 경사하강법의 적용

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수


$\Delta w_j = \eta\, \Sigma_i \big(y^{(i)} - h(\hat{y}^{(i)})\big) x_j^{(i)}$
y
class label
실제값
bias
1
w_0
w_1
w_2
w_m
inputs
x_1
x_2
x_m
$z = \Sigma$
$h(z)$
$\hat{y}$
output
예측값
$\hat{y}$
output
예측값
$z = w^T x$
$\hat{y} = h(z)$
$=$
$identity(z)$