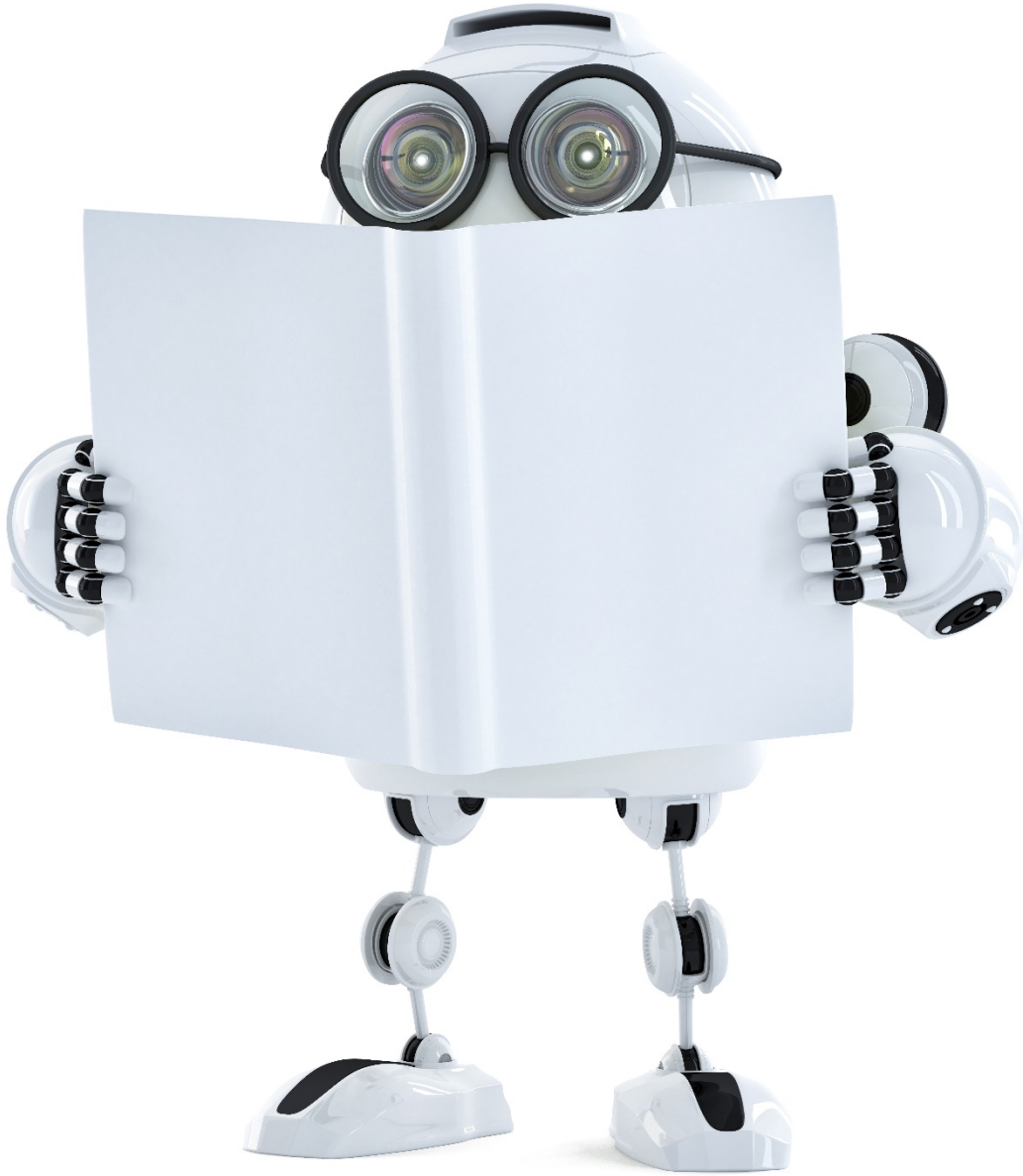
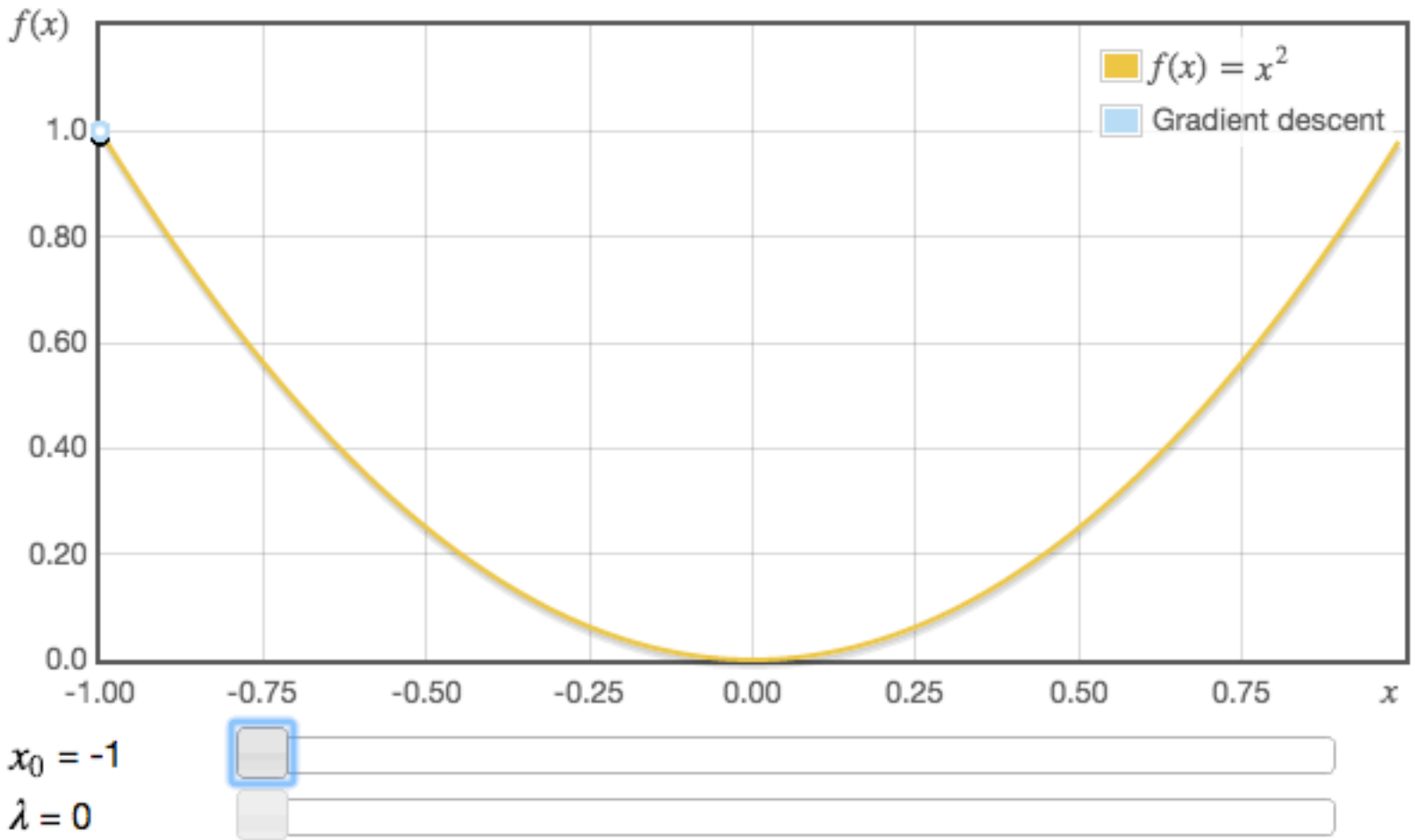
Gradient Descent

Linear Regression

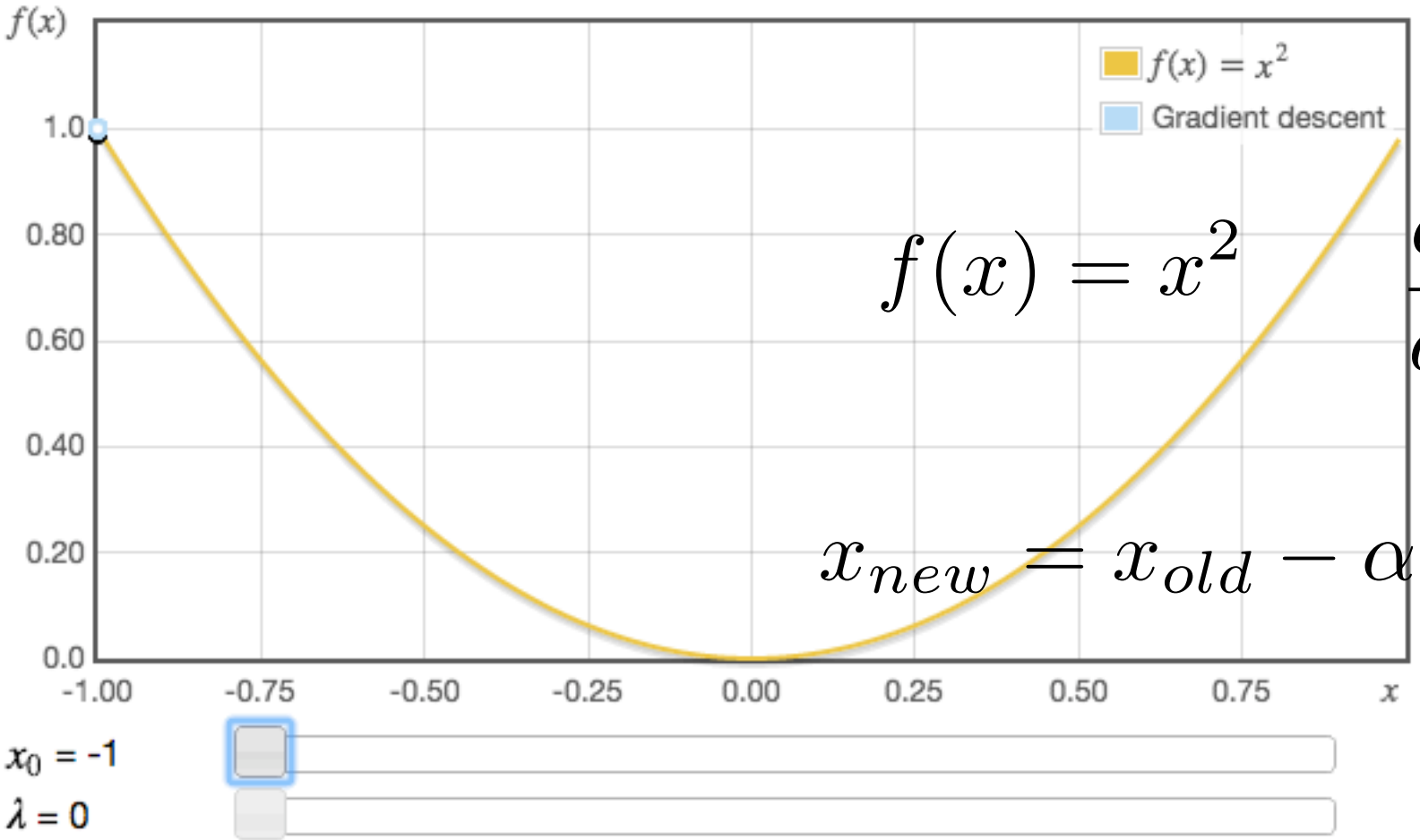


Director of TEAMLAB
Sungchul Choi

컴퓨터에 x^2 의 최적값 찾기



컴퓨터에 x^2 의 최적값 찾기



$$f(x) = x^2$$

$$\frac{dy}{dx} = 2x$$

$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

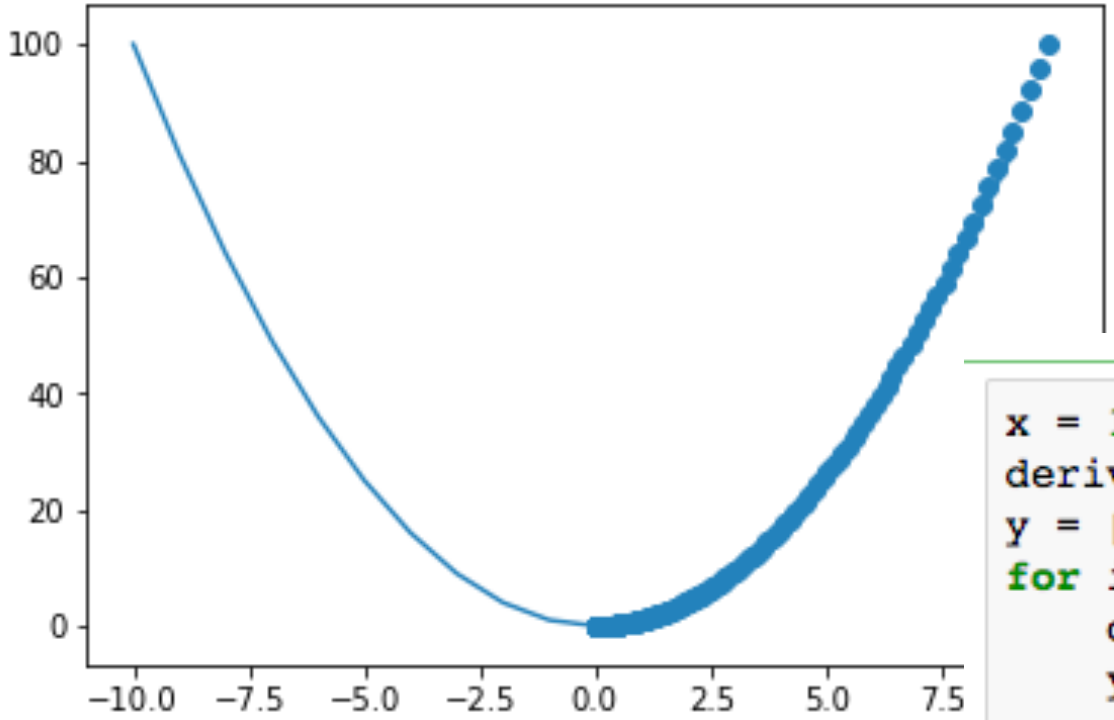
컴퓨터에 x^2 의 최적값 찾기

$$f(x) = x^2 \qquad \frac{dy}{dx} = 2x$$

$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

$$\begin{bmatrix} 1 \\ 1 - 0.1 * 2 * 1 = 0.8 \\ 0.8 - 0.1 * 2 * 0.8 = 0.64 \\ 0.64 - 0.1 * 2 * 0.64 = 0.512 \\ \vdots \end{bmatrix}$$

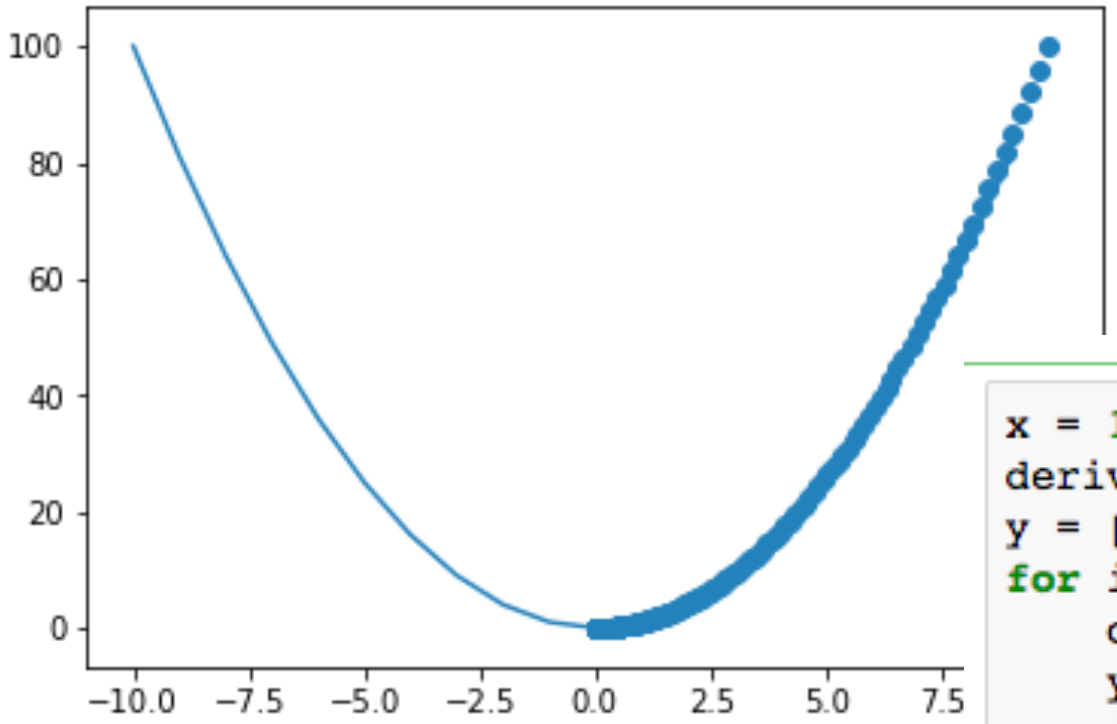
컴퓨터에 x^2 의 최적값 찾기




```
x = 10
derivative = []
y = []
for i in range(1000):
    old_value = x
    y.append(old_value ** 2)
    derivative.append(old_value - 0.01 * 2 * old_value)
    x = old_value - 0.01 *2* old_value
```

$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

컴퓨터에 x^2 의 최적값 찾기




```
x = 10
derivative = []
y = []
for i in range(1000):
    old_value = x
    y.append(old_value ** 2)
    derivative.append(old_value - 0.0
    x = old_value - 0.01 *2* old_value
```

$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

gradient	v
	10
20	9.8
19.6	9.604
9.604	9.50796
9.50796	9.41288
9.41288	9.318752
9.318752	9.225564
9.225564	9.133308
9.133308	9.041975
9.041975	8.951556
8.951556	8.86204
8.86204	8.77342
8.77342	8.685685
8.685685	8.598829
8.598829	8.51284
8.51284	8.427712
8.427712	8.343435

정해야 하는 것

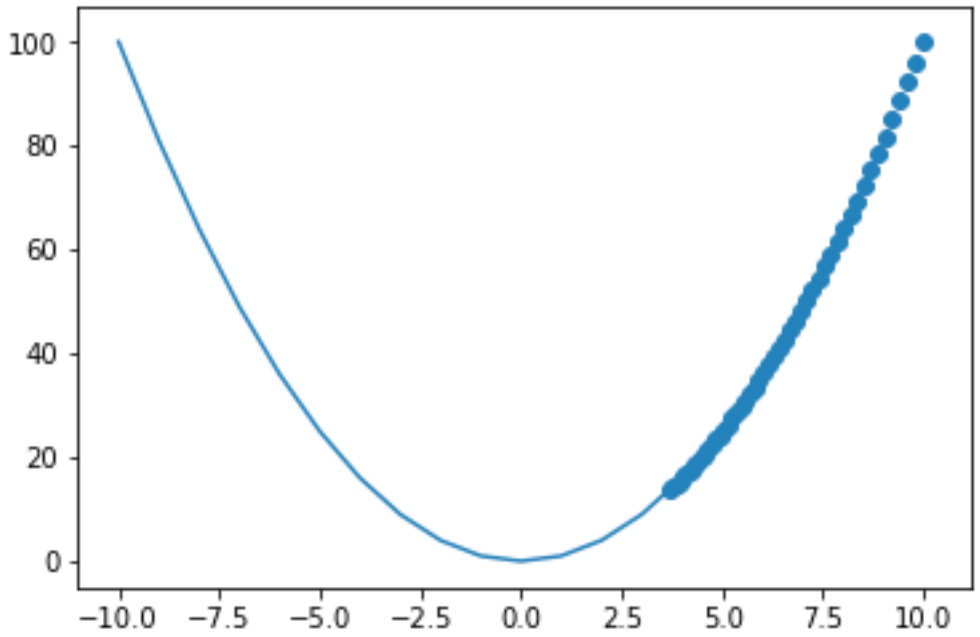
$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

Learning rate에 대한 선정

```
x = 10
derivative = []
y = []
for i in range(1000):
    old_value = x
    y.append(old_value ** 2)
    derivative.append(old_value - 0.01 * 2 * old_value)
    x = old_value - 0.01 *2* old_value
```

얼마나 많이 loop을 돌 것인가?

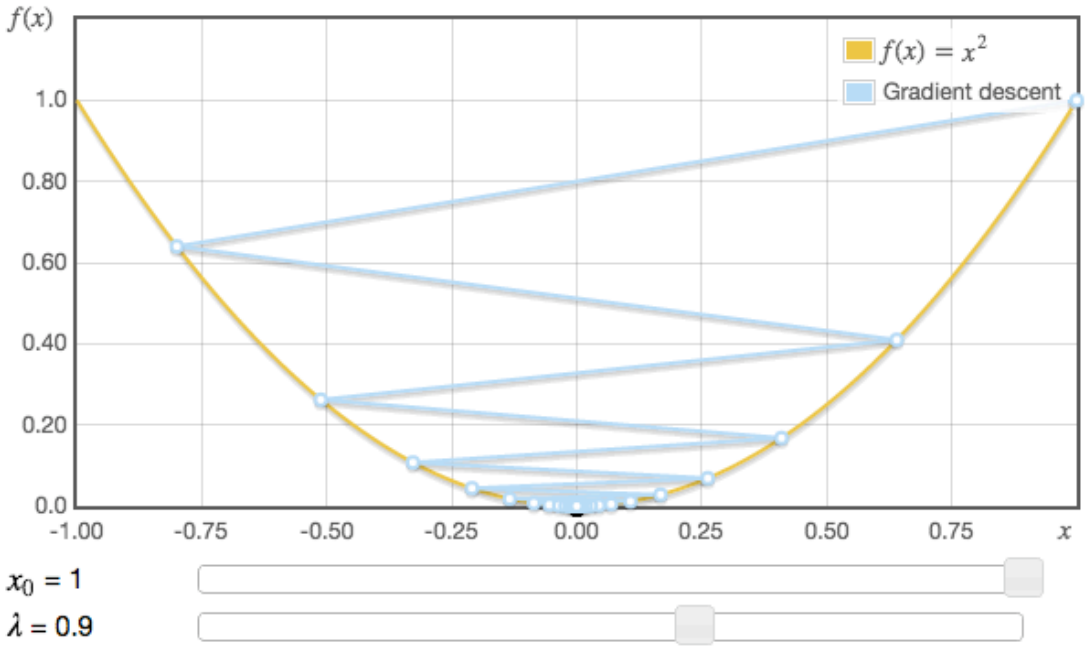
너무 작을 경우



$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

끝까지 못감
시간이 오래 걸림

너무 클 경우

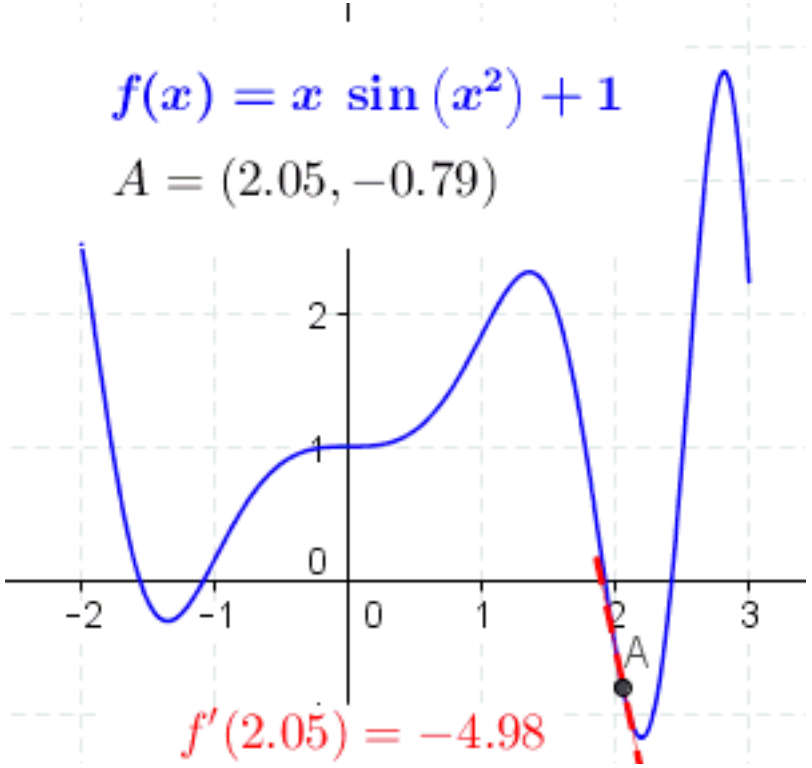


$$x_{new} = x_{old} - \alpha \times (2x_{old})$$

데이터가 튀는 문제가 생김
수렴하지 못하는 경우생김

http://www.onmyphd.com/?p=gradient.descent

굴곡이 많은 함수의 경우는?



$$\frac{df(x)}{dx} = \sin(x^2) + 2x^2\cos(x^2)$$

$$x := x - \alpha \times \sin(x^2) + 2x^2\cos(x^2)$$

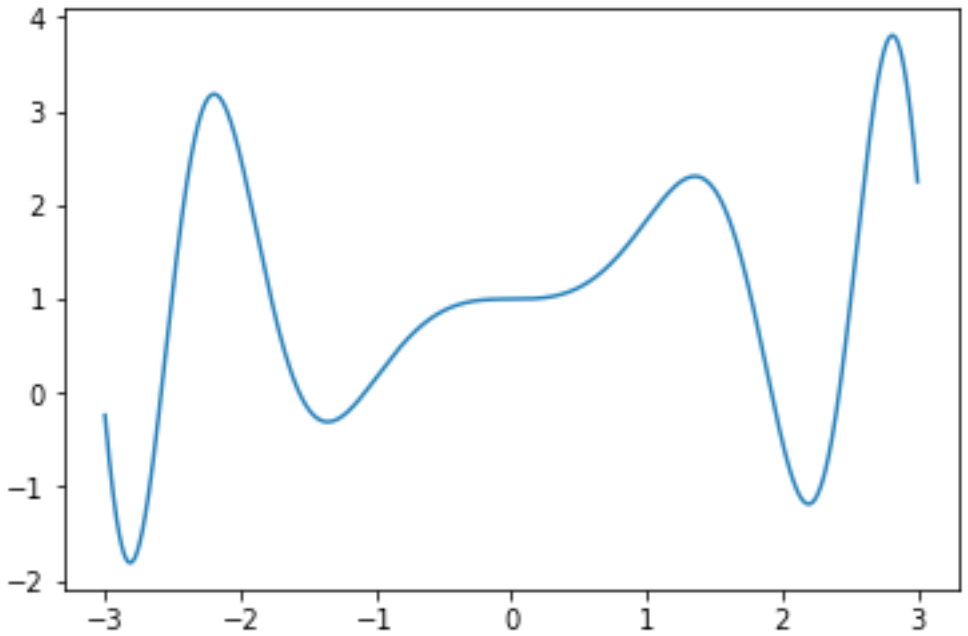
```
def sin_function(x):
    return x * np.sin(x ** 2) + 1

def derivitive_f(x):
    return np.sin(x**2) + 2 * (x **2) * np.cos(x ** 2)
```

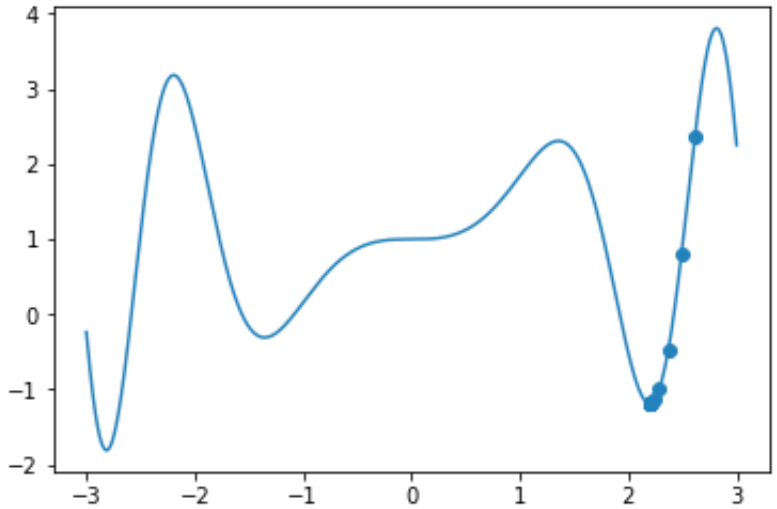
http://www.wolframalpha.com/input/?i=derivative+x+sin(x%5E2)+%2B+1

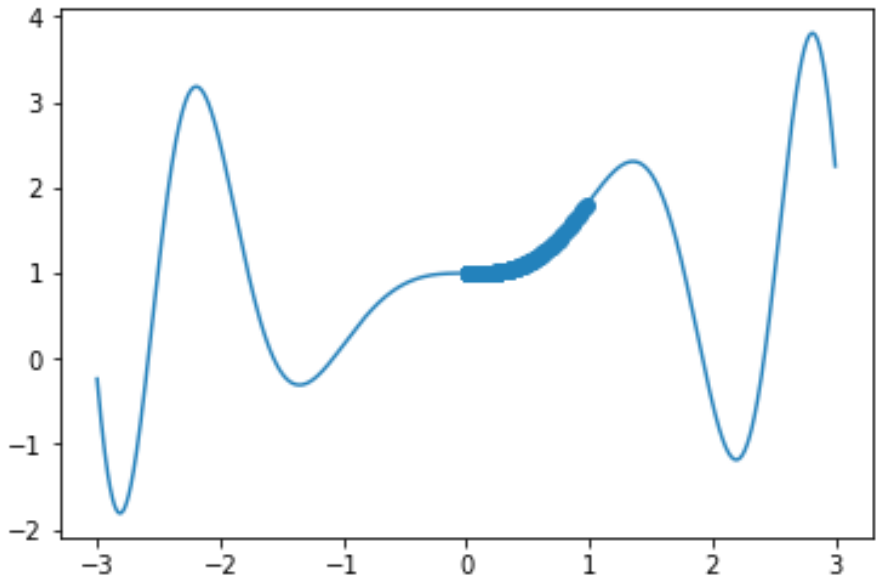
굴곡이 많은 함수의 경우는?

```
x= np.arange(-3,3,0.001)
f_x = sin_function(x)
```

```
plt.plot(x, f_x)
plt.show()
```







시작점에 따라
다른 최적값을 찾는다

Human knowledge belongs to the world.