

## 第3回：2次元のデータ Appendix2

尚 晋  
大学院経済学研究科 助教

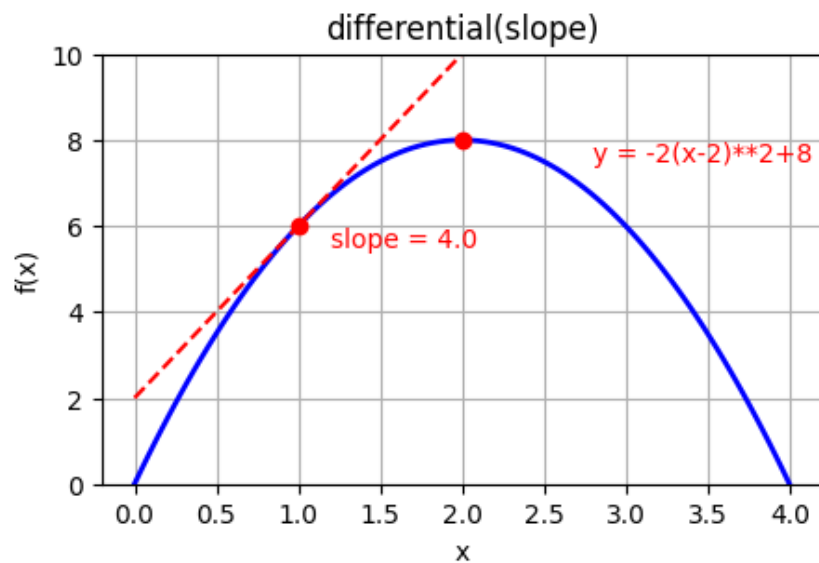
2025年5月13日

### ポイント

1. 導関数による関数の最大化と二変数関数の偏微分について

### 1 導関数による関数の最大化について

図A-1:微分は関数の傾きを計算



変数 $x$ の関数 $y = f(x)$ について,

$$f'(x) = \frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} \quad (1)$$

を $f(x)$ の一次導関数と呼ぶ。これは $f(x)$ の、 $x$ における接線の傾き(slope)を測る。

例えば：関数  $f(x) = -2(x-2)^2 + 8$  の一次導関数は

$$f'(x) = -4x + 8 \quad (2)$$

となる。点(1,6)における接線の傾きは4である(図A-1)。

次は、この問題を考えよう。

関数  $f(x) = -2(x-2)^2 + 8$  は、 $x$  がいくらのとき最大になる？

図A-1で見ると、「山の頂上」は最大値。この時の傾き平らなので、傾きは0である。  
すなわち：条件「 $f'(x) = 0$ 」を満たす  $x$  を計算すれば、関数  $f(x)$  を最大にする  $x$  の値が分かる。

したがって、関数  $f(x) = -2(x-2)^2 + 8$  の導関数  $f'(x) = -4x + 8$  をゼロとおいて解けば、 $-4x^* + 8 = 0 \Leftrightarrow x^* = 2$  が計算できる。この関数は  $f(2) = 8$  で最大。

## 2 二変数関数の偏微分について

まず、複数の変数があるような関数（多変数関数） $Y = f(x_1, x_2, \dots, x_n)$  を微分するときに、1つの変数にだけ注目し、それ以外は定数として扱うというのが偏微分。

例えば、この多変数関数  $Y = f(x_1, x_2, \dots, x_n)$  を  $x_1$  を偏微分すると、「他の説明変数  $x_2, \dots, x_n$  の値を一定として、 $x_1$  が変化した時に  $Y$  がどの程度変化するのか」を求めることができる。

多変数関数  $Y = f(x_1, x_2, \dots, x_n)$  を  $x_1$  について偏微分すると、

$$\frac{\partial Y}{\partial x_1} = \frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_1} = f_{x_1}(x_1, x_2, \dots, x_n) \quad (3)$$

となる。

それでは、スライドの18ページの最小二乗法、係数の決め方の二乗和の式を思い出そう。  
下記の通り。

$x_i$  から予想される  $y$  の値  $bx_i + a$  と、現実の値  $y_i$  が、最も小さいへだたりをもつのが、最適な直線  $y = bx + a$  の引き方である。したがって、二乗和

$$L = \sum_{i=1}^n \{y_i - (bx_i + a)\}^2$$

を最小にする  $a, b$  の値を求める。  $L$  は、 $a, b$  の二変数関数の2次式だから、最小を求めるために  $a, b$  でそれぞれ偏微分して0とおくと…

この  $L$  は  $x_i, y_i$  が定数と見ると、 $a, b$  の二変数関数の2次式と見てもよい。したがって、 $L$  の最小を求めるために  $a, b$  でそれぞれ偏微分して0とおけば求める。

**補足 2.1** なぜこの式は最小値を持つのかについて、大まかな流れを言えば、二階微分をしてヘッセ行列を求めて、このヘッセ行列は正定値であれば、凸関数を証明することができる。この場合、この関数には最小値が存在する。

図A-1のPythonコードは下記の通り。  $f(x) = 2(x-2)^2 - 8$ を描いてみてください。

Run 1: '実行してみてください'

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from matplotlib.patches import Polygon

5 plt.rcParams['axes.unicode_minus'] = False

7 fig, ax1 = plt.subplots(figsize=(5, 3))

9 # ===== differential (slope) =====
10 x = np.linspace(0, 4, 100)
11 y = -2 *(x-2)**2+8

13 ax1.plot(x, y, 'b-', linewidth=2)
14 ax1.set_title('differential(slope)')
15 ax1.set_xlabel('x')
16 ax1.set_ylabel('f(x)')
17 ax1.grid(True)
18 ax1.set_ylim(0, 10)

20 # draw partial differential equations when x=1
21 point_x = 1
22 point_y = -2 * (point_x-2)**2+8
23 print(point_x, point_y)
24 slope = 4 # partial differential equations dy/dx = -4x+8, when x=1, slope=4
25 tangent_line = slope * (x - point_x) + point_y

27 ax1.plot(x, tangent_line, 'r--', linewidth=1.5)
28 ax1.plot([point_x], [point_y], 'ro')
29 ax1.plot(2, 8, 'ro') # Y-intercept
30 ax1.text(point_x+0.2, point_y-0.5, f'slope = {slope:.1f}', color='red')
31 ax1.text(2.6, 8, 'y = -2*(x-2)**2+8', color='red')

```