第3回：2次元のデータAppendix2

尚　晋
大学院経済学研究科　助教

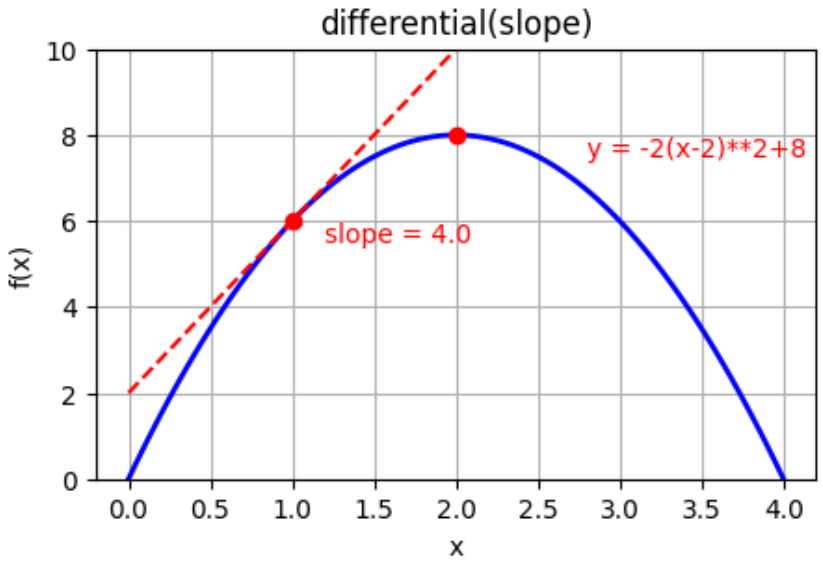
2025年5月13日

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，

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

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

$$f'(x) = -4x + 8 \tag{2}$$

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）$Y = f(x_1, x_2, ..., x_n)$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$Y = f(x_1, x_2, ..., x_n)$を$x_1$を偏微分すると、「他の説明変数$x_2, ..., x_n$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　$Y = f(x_1, x_2, ..., x_n)$を$x_1$について偏微分すると、

$$\frac{dY}{dx_1} = \frac{df(x_1, x_2, ..., x_n)}{dx_1} = f_{x_1}(x_1, x_2, ..., x_n) \tag{3}$$

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**

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

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’

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon

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

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

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

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

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

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