ＡＩ基礎セミナー

第５回
数学の基礎（２回目：解析学）

改訂履歴

日付	担当者	内容
2021/05/08	M.Takeda	Git 公開

目次

(1) はじめに

・「第５回　数学の基礎（２回目)」では、主に機械学習で使用する数学の基礎と
その Python での実装について、引き続き概観します。

・数学の基礎の２回目は、微分と偏微分などの解析学に焦点を当てます。
ニューラルネットワークモデルでは、その誤差を表す誤差関数の値を小さくするために重みを調整します。
その重み調整のアルゴリズムの一つでもある「勾配降下法（ｺｳﾊﾞｲｺｳｶﾎｳ, Gradient descent)」を紹介し、
これの理解を今回のセミナーの目標とします。

・本資料では、はじめから順番に読み進めれば、上記の目標に到達するように構成してあります。
偏微分の基本的な概念を理解することにより、ＡＩ関連の専門書が読みやすくなると思います。

・確認問題は、本文を要約したような問題となっております。
解いてみて、理解度を確認してみてください。
数学が苦手な方には、ご苦労をお掛けしますが、概念だけでも理解いただければ、と思います。

(2) 微分

・平均変化率とその極限値としての微分について、以下に記します。

(2.1) 平均変化率・微分可能・導関数・微分係数

・区間 J（例えば -100≦x≦100 等）で定義された関数 y = f(x) について、
変数 x（x ∈ J） が x から x+⊿x まで変化する時、y が y から y+⊿y まで変化したとします。
この時、x の増分⊿x に対する y の増分は ⊿y で、
⊿y/⊿x は、y = f(x) の「平均変化率（ﾍｲｷﾝｼﾞﾍﾝｶﾘﾂ、average rate of change)」といい、
次式で与えられます：

平均変化率

$$⊿y/⊿x = (f(x+⊿x) - f(x)) /⊿x \quad \cdots(式2.1\text{-}1)$$

・平均変化率について、点 x における増分⊿xを限りなく 0 に近づけた場合に、
平均変化率の有限な極限値が、区間 J 内の各点で存在する時、
関数 y = f(x) は区間 J で「微分可能（ﾋﾞﾌﾞﾝｶﾉｳ、differentiable)」である、と言います。

その極限値を、
関数 y = f(x) の「導関数（どうかんすう、derived function 又は derivative)」と呼び、
f'(x) または df(x)/dx と表現します。
また、関数 y = f(x) の 導関数を求めることを、
関数 f(x) を「微分する（differentiate)」と言います。

導関数

$$\begin{aligned} df(x)/dx &= f'(x) \\ &= \lim_{⊿x\to 0} ⊿y /⊿x \\ &= \lim_{⊿x\to 0} (f(x+⊿x) - f(x)) /⊿x \end{aligned} \quad \cdots(式2.1\text{-}2)$$

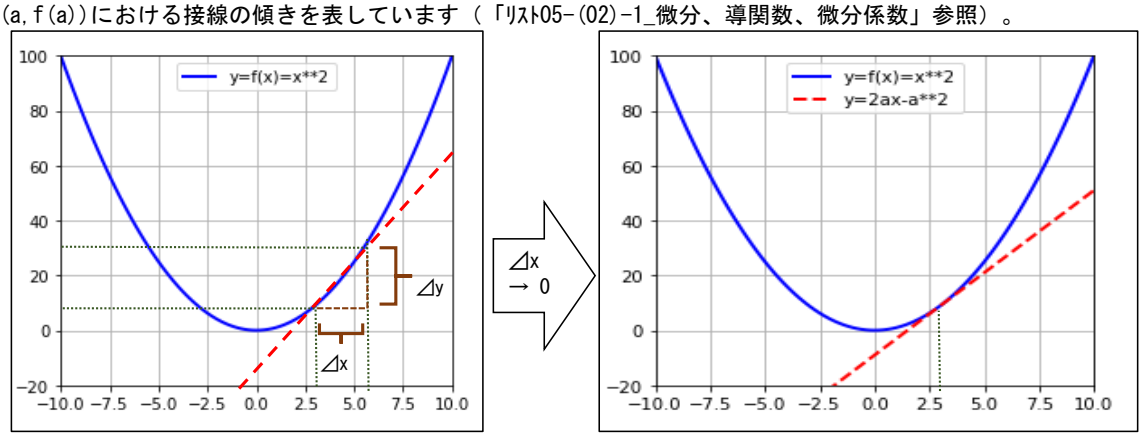
※「$\lim_{⊿x\to 0}$」は⊿xを限りなく0に近づける時の極限値

・変数 x = a における関数 y = f(x) の導関数の値 f'(a) を
関数 y = f(x) の x = a における「微分係数（ﾋﾞﾌﾞﾝｹｲｽｳ、differential coefficient)」
と呼び、f'(a) または $df(a)/dx$ と表現します。

微分係数

$$df(a)/dx = f'(a) \quad \cdots(式2.1\text{-}3)$$

・変数 x = a における微分係数 f'(a) は、y = f(x) のグラフ上では、
点(a,f(a))における接線の傾きを表しています（「ﾘｽﾄ05-(02)-1_微分、導関数、微分係数」参照）。



(x=3～5 における平均変化率＝⊿y/⊿x)　　(x=3 における微分係数＝dy/dx)

・微分可能性についての条件は、以下のとおりです。

微分可能性についての条件

・変数 x = a における関数 y = f(x) の導関数の値 f'(a) が存在する為の必要十分条件は、

右微分係数: $\lim_{\Delta x \to +0} (f(a + \Delta x) - f(a)) / \Delta x$

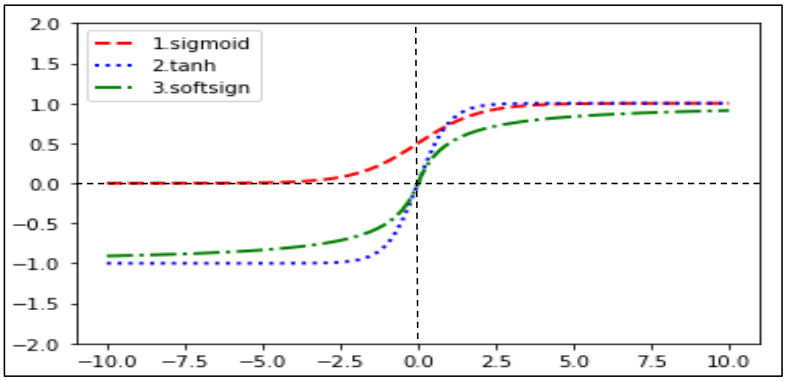
左微分係数: $\lim_{\Delta x \to -0} (f(a + \Delta x) - f(a)) / \Delta x$

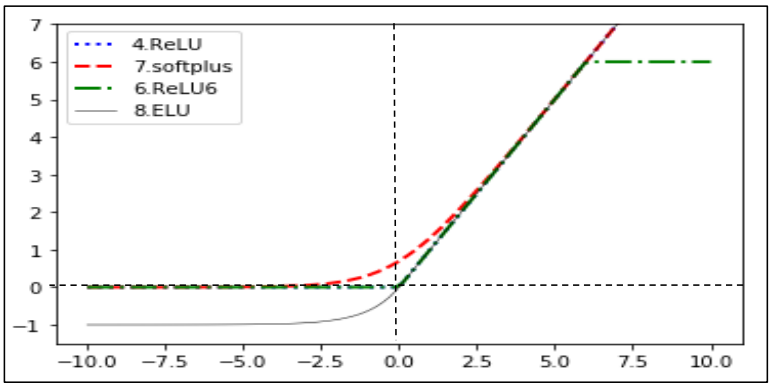
が存在して、等しいことである。

「$\lim_{\Delta x \to +0}$」は⊿xを正方向から負方向に向かって限りなく 0に近づける時の極限値

「$\lim_{\Delta x \to -0}$」は⊿xを負方向から正方向に向かって限りなく 0に近づける時の極限値

・微分可能性について、セミナー第2回目に触れた活性化関数のいくつかについて見てみましょう。





1 ｼｸﾞﾓｲﾄﾞ関数

$\sigma(x) = 1 / \{ 1 + \exp(-x) \}$

2 ﾊｲﾊﾟﾎﾞﾘｯｸﾀﾝｼﾞｪﾝﾄ（双曲線正接関数）

$\tanh(x) = \{\exp(x) - \exp(-x)\} / \{\exp(x) + \exp(-x)\}$

3 ソフトサイン関数

$\mathrm{softsign}(x) = x / \{\mathrm{abs}(x) + 1\}$

4 正規化線形関数（ランプ関数）

$\mathrm{ReLU}(x) = \max(0, x)$

6 正規化線形関数（ランプ関数、上限＝6）

$\mathrm{ReLU6}(x) = \min(\max(0, x), 6)$

7 ソフトプラス関数

$\mathrm{softplus}(x) = \log(\exp(x) + 1)$

8 Exponential Linear Unit

$\mathrm{ELU}(x) = \exp(x) - 1 \quad \text{for } x < 0, \quad x \quad \text{for } x \geq 0$

1,2,3,7,8 はxの全領域で微分可能

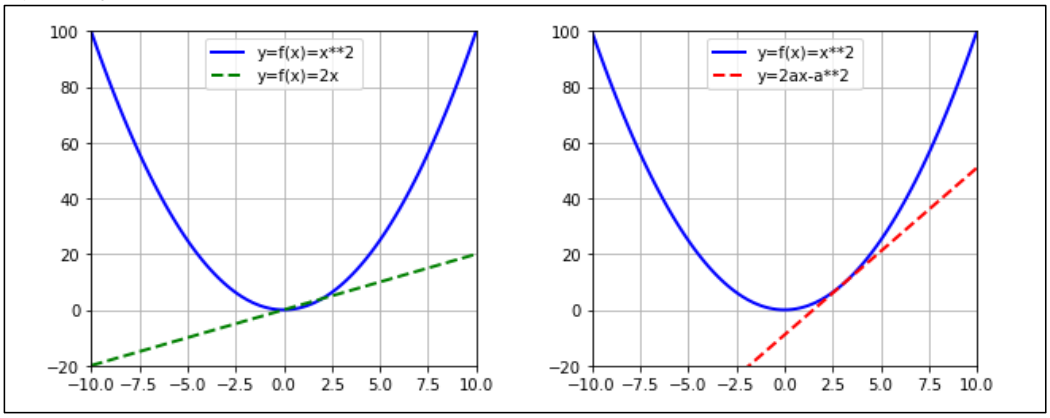
4 は x=0 で微分可能でなく、それ以外の領域では微分可能

6 は x=0,6 で微分可能でなく、それ以外の領域では微分可能

(微分、導関数、微分係数の実装例)

```
#************************************************
# ﾘｽﾄ05-(02)-1_微分、導関数、微分係数
#************************************************
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# 入力データを準備 ------------------------------
X_min = -10              # X の下限（表示用）
X_max = 10               # X の上限（表示用）
Y_min = -20              # Y の下限（表示用）
Y_max = X_min * X_min    # Y の上限（表示用）
Xa = 3                   # 接線表示X座標

# X方向の表示範囲とピッチ
X_pitch = 0.2            # X方向のデータのピッチ
X_cnt = (int)((X_max - X_min) / X_pitch + 1)
Xlist = np.zeros([X_cnt]) # X座標値のリスト
for xi in range(0, X_cnt):
    Xlist[xi] = X_min + (xi * X_pitch)

# 元の関数 y = f(x) = x**2 ----------------------
def yEqualsSquareX(xlist):
    ylist =  xlist * xlist
    return ylist

# 導関数 y = f'(x) = 2x -------------------------
def yEquals2x(xlist):
    ylist =  2 * xlist
    return ylist

# x=a における接線関数 y = 2ax - a**2 -----------
def yTangentx(xlist, a):
    ylist =  2*a*xlist - a**2
    return ylist

# メイン ----------------------------------------

# Y座標値のリスト
Ylist = yEqualsSquareX(Xlist)        # 元関数 y = f(x) = x**2
YdashList = yEquals2x(Xlist)         # 導関数 y = f'(x) = 2x
YTangentList = yTangentx(Xlist, Xa)  # x=a における接線関数 y = 2ax - a**2

# グラフ描画を複数並べる為の準備
plt.figure(figsize=(10, 4))
plt.subplots_adjust(wspace=0.3)

# 元関数(y = f(x) = x**2)と、導関数(y = f'(x) = 2x)をグラフ表示
plt.subplot(1, 2, 1)
plt.plot(Xlist, Ylist, color='blue', label='y=f(x)=x**2', linewidth=2)
plt.plot(Xlist, YdashList, color='green', label='y=f''(x)=2x', linestyle='--', linewidth=2)
plt.xlim(X_min, X_max)
plt.ylim(Y_min, Y_max)
plt.legend(loc='upper center')
plt.grid(True)

# 元関数(y = f(x) = x**2)と、x=a における接線(y = 2ax - a**2)をグラフ表示
plt.subplot(1, 2, 2)
plt.plot(Xlist, Ylist, color='blue', label='y=f(x)=x**2', linewidth=2)
plt.plot(Xlist, YTangentList, color='red', label='y=2ax-a**2', linestyle='--', linewidth=2)
plt.xlim(X_min, X_max)
plt.ylim(Y_min, Y_max)
plt.legend(loc='upper center')
plt.grid(True)

plt.show()
```

（実行結果）



【出典・参考】

微分⇒ https://ja.wikipedia.org/wiki/微分

微分⇒ 「解析概論」高木貞治著 岩波書店

微分可能性⇒ 「数学公式事典」 黒田孝朗・須田貞之著 文研出版 1978

(2.2) 導関数の計算例

・以下に、導関数の計算例を幾つか示します：

(1) 定数関数：f(x) = C

定数関数：f(x) = C　(Cは定数)の導関数

$$dC/dx = 0 \qquad \cdots(式2.2\text{-}1)$$

（公式の導出）

$$df(x)/dx = \lim_{h\to 0} (f(x+h) - f(x)) / h$$
$$= \lim_{h\to 0} (C - C) / h = 0$$

(2) x の n次式：f(x) = x^n

x の n次式：f(x) = x^n　の導関数

$$dx^n/dx = n\,x^{n-1} \qquad \cdots(式2.2\text{-}2)$$

（公式の導出）

$$dx^n/dx =_{※0} \lim_{\Delta x\to 0} ((x+\Delta x)^n - x^n) / \Delta x$$
$$=_{※1} \lim_{h\to 0} ((x+h)^n - x^n) / h$$
$$=_{※2} \lim_{h\to 0} (({}_nC_0 x^n h^0 + {}_nC_1 x^{n-1}h^1 + \cdots + {}_nC_k x^{n-k}h^k + \cdots + {}_nC_n x^0 h^n) - x^n) / h$$
$$=_{※3} \lim_{h\to 0} ((x^n h^0 + {}_nC_1 x^{n-1}h^1 + \cdots + {}_nC_k x^{n-k}h^k + \cdots + x^0 h^n) - x^n) / h$$
$$=_{※4} \lim_{h\to 0} ((x^n + {}_nC_1 x^{n-1}h^1 + \cdots + {}_nC_k x^{n-k}h^k + \cdots + h^n) - x^n) / h$$
$$= \lim_{h\to 0} ({}_nC_1 x^{n-1}h^1 + \cdots + {}_nC_k x^{n-k}h^k + \cdots + h^n) / h$$
$$=_{※5} \lim_{h\to 0} ({}_nC_1 x^{n-1} + \cdots + {}_nC_k x^{n-k}h^{k-1} + \cdots + h^{n-1}) \; h / h$$
$$= \lim_{h\to 0} ({}_nC_1 x^{n-1} + \cdots + {}_nC_k x^{n-k}h^{k-1} + \cdots + h^{n-1})$$
$$=_{※6} {}_nC_1 x^{n-1}$$
$$=_{※7} nx^{n-1}$$

※0：「$\lim_{\Delta x\to 0}$」はΔxを限りなく0に近づける時の極限値
※1：Δxをhで置換
※2：二項定理「 $(x+h)^n = \Sigma_{k=0}^{n} {}_nC_k x^{n-k}h^k$ 」を適用
※3：${}_nC_n=1$、${}_nC_0=1$ を適用
※4：$x^0=1$、$h^0=1$ を適用
※5：$h^1=h$、$h^k=h^{k-1}h^1=h^{k-1}h$ を適用
※6：h が係数になっている項を0で評価
※7：${}_nC_1=n$ を適用

（注1）上記の証明方法以外に、数学的帰納法による証明もあります。
（注2）上記の証明は、n は自然数であることを想定したものですが、
実は、n が任意の実数の場合にも成立します。

（例）

$$dx/dx =_{※8} dx^1/dx = 1x^0 = 1$$
$$dx^2/dx = 2x$$
$$dx^3/dx = 3x^2$$
$$dx^4/dx = 4x^3$$

※8：$x^0=1$、$x^1=x$ を適用

【出典・参考】
⇒ 「数学公式事典」 黒田孝朗・須田貞之著 文研出版 1978

(2.3) 導関数についての公式

・f(x)、g(x) を微分可能な関数とするとき、以下の公式があります。

(1) 関数の和・差・商・積の微分

・和・差の微分　　$\{ f(x) \pm g(x) \}' = f'(x) \pm g'(x)$　　…(式2.3-1)

・ｽｶﾗｰ倍の微分　　$\{ \lambda f(x) \}' = \lambda f'(x)$　　（λ はｽｶﾗｰ）　　…(式2.3-2)

・積の微分　　$\{ f(x) \cdot g(x) \}' = f'(x) \cdot g(x) + f(x) \cdot g'(x)$　　…(式2.3-3)

・商の微分　　$\{ f(x) / g(x) \}' = \{ f'(x) \cdot g(x) - f(x) \cdot g'(x) \} / \{ g(x) \}^2$　　…(式2.3-4)

・商の微分(分子=1)　　$\{ 1 / g(x) \}' = -g'(x) / \{ g(x) \}^2$　　…(式2.3-5)

（例）

・和・差・ｽｶﾗｰ倍の微分

$y = ax^2 - bx + c$

$$\frac{dy}{dx} = \frac{d(ax^2)}{dx} + \frac{d(-bx)}{dx} + \frac{dc}{dx} = 2ax - b + 0 = 2ax - b$$

・積の微分

$y = (ax + b)(cx + d)$

$$\frac{dy}{dx} = \frac{d(ax + b)}{dx}(cx + d) + (ax + b)\frac{d(cx + d)}{dx} = a(cx + d) + (ax + b)c = (acx + ad) + (acx + bc) = 2acx + ad + bc$$

・商の微分

$y = (ax + b) / (cx + d)$

$$\frac{dy}{dx} = \left\{ \frac{d(ax + b)}{dx}(cx + d) - (ax + b)\frac{d(cx + d)}{dx} \right\} / (cx + d)^2 = \{ a(cx + d) - (ax + b)c \} / (cx + d)^2 = (ad - bc) / (cx + d)^2$$

・商の微分(分子=1)

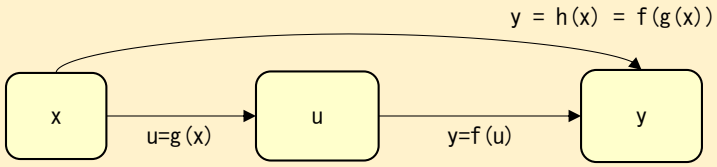
$y = 1 / (cx + d)$

$$\frac{dy}{dx} = -\frac{d(cx + d)}{dx} / (cx + d)^2 = \{ a(cx + d) - (ax + b)c \} / (cx + d)^2 = -c / (cx + d)^2$$

(2) 合成関数の微分

・合成関数の微分

y が変数 u の関数 y = f(u) で、
u が変数 x の関数 u = g(x) の時、
y は変数 x の関数 y = h(x) となります。



関数h は、関数g と関数f との「合成関数（ごうせいかんすう、composite function)」
と言い、次のように表現します。

$$y = h(x) = f(g(x))$$

関数g と関数f とが微分可能である時、
関数g と関数f との合成関数h の導関数は、
それぞれの導関数の積で与えられます。
これを「連鎖律（れんさりつ、chain rule)」と言います。

$$\begin{aligned} dy/dx &= dh(x)/dx \\ &= df(g(x))/dx \\ &= df/dg \cdot dg/dx \\ &= dy/du \cdot du/dx \end{aligned} \qquad \cdots(式2.3\text{-}6)$$

この連鎖律は、多重の合成関数にも拡張することが出来ます。
（例） y = f(g(h(x))) の時、

$$dy/dx = df/dg \cdot dg/dh \cdot dh/dx \qquad \cdots(式2.3\text{-}7)$$

（例）

y = (ax + b)^3 の微分を合成関数の微分法により求めてみます。

y = f(g(x)) = g(x)^3, g(x) = ax + b として、

$$\begin{aligned} dy/dx &=_{※1} df/dg \cdot dg/dx \\ &=_{※2} 3g^2 \cdot a \\ &=_{※3} 3(ax + b)^2 \cdot a \\ &= 3(a^2x^2 + 2abx + b^2) \cdot a \\ &= 3a^3x^2 + 6a^2bx + 3ab^2 \end{aligned}$$

∴

$$d(ax+b)^3/dx = 3a^3x^2 + 6a^2bx + 3ab^2$$

※1：合成関数の微分法
※2：$df/dg=3g^2$、$dg/dx=a$
※3：g(x) = ax + b

【出典・参考】
合成関数⇒ https://ja.wikipedia.org/wiki/写像の合成
連鎖律⇒ https://ja.wikipedia.org/wiki/連鎖律

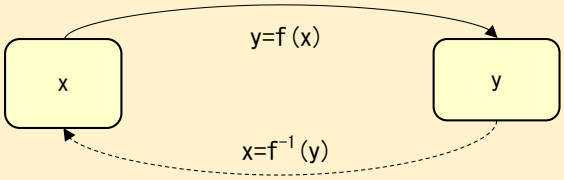
(3) 逆関数の微分

・逆関数の微分

x の関数 $y = f(x)$ が1対1の写像である時、
関数 $y = f(x)$ の「逆関数（ぎゃくかんすう、inverse function)」が存在し、
次のように表現します。

$$x = f^{-1}(y)$$

逆関数について、$f^{-1}(f(x)) = x$ が成立します。



関数 $y = f(x)$ が微分可能で、$df(x)/dx \neq 0$ ならば、
その逆関数 $x = f^{-1}(y)$ もまた微分可能で、その導関数は次式で与えられます。

$$dg(y)/dy = 1 \,/\, df(x)/dx \qquad \cdots(式2.3\text{-}8)$$

又は

$$dx/dy = 1 \,/\, dy/dx \qquad \cdots(式2.3\text{-}9)$$

なお、関数は一般に従属変数y を独立変数x の関数として表現するので、
逆関数もそのように書きかえて(x→y, y→x に書きかえて)扱い、
逆関数の微分は、次式で与えられます：

$$dy/dx = 1 \,/\, dx/dy \qquad \cdots(式2.3\text{-}10)$$

（例）

$y = \sin^{-1}(x)$ の導関数を、逆関数の微分から求めてみます。

$y=\sin^{-1}(x)$ の逆関数は $x=\sin(y)$ で、
$-\pi/2<y<\pi/2$ の範囲で $x=\sin(y)$ は単調増加なので $(-1<x<1)$、
その傾きである微分係数も正値となり、

$$dx/dy = d\sin(y)/dy > 0$$

となります。
よって、

$$d(\sin^{-1}(x))/dx$$
$$= dy/dx$$
$$=_{※1} 1/dx/dy \qquad ※1：逆関数の微分の関係式を適用$$
$$=_{※2} 1/\cos(y) \qquad ※2：dx/dy = d(\sin(y))/dy = \cos(y)$$
$$=_{※3} 1/\sqrt{1 - \sin^2(y)} \qquad ※3：1=\cos^2(y)+\sin^2(y) \text{ の関係式を適用}$$
$$= 1/\sqrt{1 - x^2} \quad (-1<x<1)$$

∴

$$d(\sin^{-1}(x))/dx = 1/\sqrt{1 - x^2} \quad (-1<x<1)$$

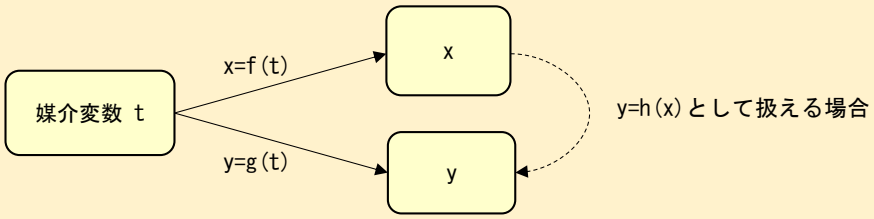
【出典・参考】

逆関数⇒ https://kou.benesse.co.jp/nigate/math/a14m2102.html
逆関数の微分⇒ http://naop.jp/text/3/bibun/bibun4.html
逆関数の微分⇒ http://www.rirurara.com/cfv21/math/invderiv.htm
逆関数の微分⇒ https://detail.chiebukuro.yahoo.co.jp/qa/question_detail/q1226107652

(4) 媒介変数で表された関数の微分

・媒介変数tで表された関数の微分

x が媒介変数 t の関数 x = f(t) で、
y が媒介変数 t の関数 y = g(t) で、各々表されているとき、
y を x の関数として扱える場合があります。



関数 x=f(t)、y=g(t) が微分可能で、$dx/dt = df(t)/dt \neq 0$ ならば、
y を x の関数とみなすことが出来て、
y の x による導関数は、次式で与えられます：

$$dy/dx = (dy/dt)/(dx/dt) \quad \cdots(式2.3\text{-}11)$$
$$= (dg(t)/dt)/(df(t)/dt)$$
$$= g'(t)/f'(t)$$

（例）

媒介変数θ $(0<\theta<\pi)$の2つの関数

$$x(\theta) = \theta - \sin(\theta)$$
$$y(\theta) = 1 - \cos(\theta)$$

の導関数は

$$dx(\theta)/d\theta = 1 - \cos(\theta)$$
$$dy(\theta)/d\theta = \sin(\theta)$$

で、$(0<\theta<\pi)$の範囲で

$$dx(\theta)/d\theta > 0$$

となる。

従って$(0<\theta<\pi)$の範囲で y を x の関数とみなすことが出来て、その導関数は

$$dy/dx = dy/d\theta \,/\, dx/d\theta$$
$$= \sin(\theta)/\{1 - \cos(\theta)\}$$

(3) 基本的な関数とその微分

・基本的な関数とその導関数を以下に記します。

(3.1) べき関数

(1) べき関数とは

べき関数

・「べき関数（power function)」とは、xを変数、nを自然数としたとき
xを n回繰り返し掛けたもので、以下の様に表現します。
べき関数で繰り返し掛ける回数 n は、関数の「べき指数（ﾍﾞｷｼｽｳ, exponent)」と呼びます。

$$y(x) = x^n = \underbrace{x \cdot x \cdot \cdots \cdot x}_{n\ 回} \quad (n：自然数のべき指数) \quad \cdots(式3.1\text{-}1)$$

・繰り返し回数 nを実数αに拡張したものも、一般に「べき関数」と言います。

$$y(x) = x^{\alpha} \quad (\alpha は実数のべき指数) \quad \cdots(式3.1\text{-}2)$$

（例）

・べき指数＝ 2 のべき関数

$$y(x) = x^2$$
$$y(10) = 10^2 = 10 \cdot 10 = 100$$

・べき指数＝ -1 のべき関数

$$y(x) = x^{-1}$$
$$y(10) = 10^{-1} = 1 / 10^1 = 1 / 10 = 0.1$$

(2) べき関数の性質

べき関数の性質を幾つか列挙しておきます

・積
$$x^m \cdot x^n = x^{m+n} \quad (m,n：任意実数) \quad \cdots(式3.1\text{-}3)$$
・商
$$x^m / x^n = x^{m-n} \quad (x \neq 0,\ m,n：任意実数) \quad \cdots(式3.1\text{-}4)$$
・べき乗
$$(x^m)^n = x^{m \cdot n} \quad (m,n：任意実数) \quad \cdots(式3.1\text{-}5)$$
・積のべき乗
$$(x \cdot y)^n = x^n \cdot y^n \quad (n：任意実数) \quad \cdots(式3.1\text{-}6)$$
・商のべき乗
$$(x / y)^n = x^n / y^n \quad (y \neq 0,\ n：任意実数) \quad \cdots(式3.1\text{-}7)$$

(3) べき関数の導関数

・「べき関数 (power function)」の導関数は、次式により与えられます：

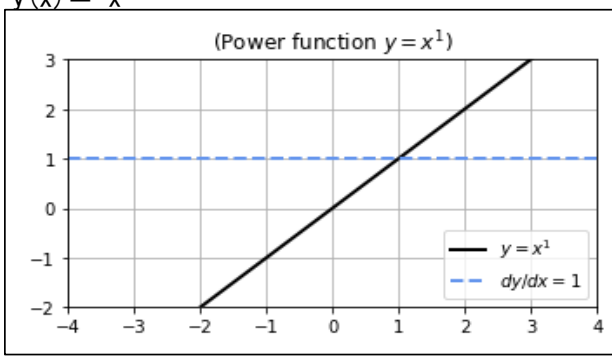
$$d\ x^{\alpha}/dx \quad = \quad \alpha x^{\alpha-1} \qquad (\alpha は実数) \qquad \cdots(式3.1\text{-}8)$$

(4) べき関数とその導関数のグラフ例

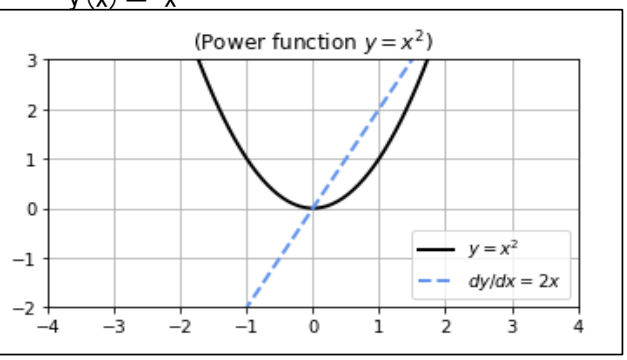
（「(5) べき関数とその導関数の実装例」を参照）

(凡例)　黒実線： 関数
青点線： その導関数

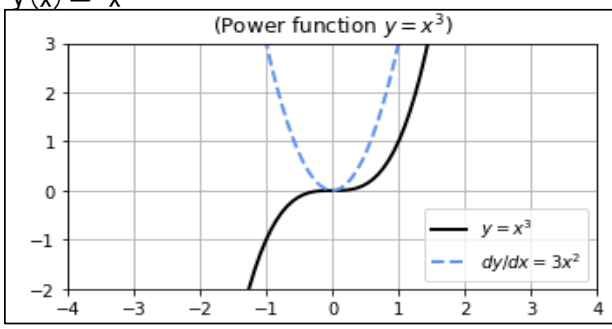
$y(x) = x^1$



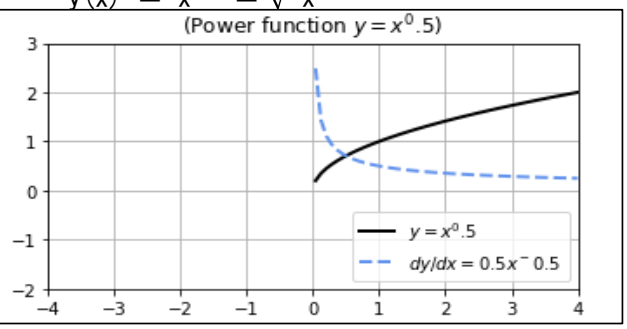
$y(x) = x^2$



$y(x) = x^3$



$y(x) = x^{0.5} = \sqrt{x}$



$(x \geqq 0)$

(5) べき関数とその導関数の実装例

```
#**************************************************
# ﾘｽﾄ05-(03)-1_べき関数とその導関数の実装例
#**************************************************
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# べき関数
def fPower(x, a):
    return x**a

# べき関数の微分
def dfPower(x, a):
    return a * x**(a-1)

# x データ列
x = np.linspace(-4, 4, 100)
x2 = np.linspace(0.01, 4, 100)

# べき関数とその導関数
y1 = fPower(x, 1)
y2 = fPower(x, 2)
```

```
y3 = fPower(x, 3)
y4 = fPower(x2, 0.5)
dy1 = dfPower(x, 1)
dy2 = dfPower(x, 2)
dy3 = dfPower(x, 3)
dy4 = dfPower(x2, 0.5)

# グラフ表示
plt.figure(figsize=(12,7))
plt.subplots_adjust(wspace=0.2, hspace=0.5)

# グラフ表示：べき関数 y=x^1
plt.subplot(2,2,1)
plt.title("(Power function $y=x^1$)")
plt.plot(x, y1, 'black', linewidth=2, label='$y=x^1$')
plt.plot(x, dy1, 'cornflowerblue', linewidth=2, label='$dy/dx=1$', linestyle='--')
plt.ylim(-2, 3)
plt.xlim(-4, 4)
plt.grid(True)
plt.legend(loc='lower right')

# グラフ表示：べき関数 y=x^2
plt.subplot(2,2,2)
plt.title("(Power function $y=x^2$)")
plt.plot(x, y2, 'black', linewidth=2, label='$y=x^2$')
plt.plot(x, dy2, 'cornflowerblue', linewidth=2, label='$dy/dx=2x$', linestyle='--')
plt.ylim(-2, 3)
plt.xlim(-4, 4)
plt.grid(True)
plt.legend(loc='lower right')

# グラフ表示：べき関数 y=x^3
plt.subplot(2,2,3)
plt.title("(Power function $y=x^3$)")
plt.plot(x, y3, 'black', linewidth=2, label='$y=x^3$')
plt.plot(x, dy3, 'cornflowerblue', linewidth=2, label='$dy/dx=3x^2$', linestyle='--')
plt.ylim(-2, 3)
plt.xlim(-4, 4)
plt.grid(True)
plt.legend(loc='lower right')

# グラフ表示：べき関数 y=x^0.5
plt.subplot(2,2,4)
plt.title("(Power function $y=x^0.5$)")
plt.plot(x2, y4, 'black', linewidth=2, label='$y=x^0.5$')
plt.plot(x2, dy4, 'cornflowerblue', linewidth=2, label='$dy/dx=0.5x^-0.5$', linestyle='--')
plt.ylim(-2, 3)
plt.xlim(-4, 4)
plt.grid(True)
plt.legend(loc='lower right')

plt.show()
```

(3.2) 指数関数

(1) 指数関数とは

指数関数

・「指数関数（ｼｽｳｶﾝｽｳ, exponential function)」とは、
「べき関数」の「べき指数」を変数 xとしたもので、
$y = a^x$ $(a > 0,\ a \neq 1)$で表されます。
このような関数のことを、「a を底とする x の指数関数」といいます。

$$y(x) = a^x \quad (a>0,\ a\neq 1) \qquad \cdots(式3.2\text{-}1)$$
$$= \underbrace{a \cdot a \cdot \cdots \cdot a}_{x回}$$

（※ xは実数でも成り立ちます）

（例）

・10 を底とする x の指数関数

$y(x) = 10^x$

$y(2) = 10^2 = 10 \cdot 10 = 100$
$y(1) = 10^1 = 10 = 10$
$y(0) = 10^0 = 1 = 1$
$y(-1) = 10^{-1} = 1 / 10^1 = 1 / 10 = 0.1$
$y(-2) = 10^{-2} = 1 / 10^2 = 1 / 100 = 0.01$

(2) 指数関数の性質

指数関数の性質を幾つか列挙しておきます

・指数が0
$a^0 = 1$ $(a>0)$ …(式3.2-2)

・指数が負
$a^{-n} = 1/a^n$ $(a>0,\ n>0)$ …(式3.2-3)

・底が同じ指数の積
$a^m \cdot a^n = a^{m+n}$ $(a>0,\ m,n$：任意実数$)$ …(式3.2-4)

・底が同じ指数の商
$a^m / a^n = a^{m-n}$ $(a>0,\ m,n$：任意実数$)$ …(式3.2-5)

・指数のべき乗
$(a^m)^n = a^{m \cdot n}$ $(a>0,\ m,n$：任意実数$)$ …(式3.2-6)

・積のべき乗
$(a \cdot b)^n = a^n \cdot b^n$ $(a,b>0,\ n$：任意実数$)$ …(式3.2-7)

・商のべき乗
$(a / b)^n = a^n / b^n$ $(a,b>0,\ n$：任意実数$)$ …(式3.2-8)

【出典・参考】

指数関数⇒ https://sci-pursuit.com/math/exponential-function.html
指数関数⇒ https://ja.wikipedia.org/wiki/指数関数
⇒ 「数学公式事典」 黒田孝朗・須田貞之著 文研出版 1978

(3) 指数関数の導関数

指数関数の導関数

・aを底とする指数（base a exponential）

$$\frac{d}{dx} a^x = a^x \log(a) \qquad (a>0, a\neq 1) \qquad \cdots(式3.2\text{-}9)$$

・eを底とする指数（natural exponential）

$$\frac{d}{dx} e^x = e^x \qquad (※1) \qquad \cdots(式3.2\text{-}10)$$

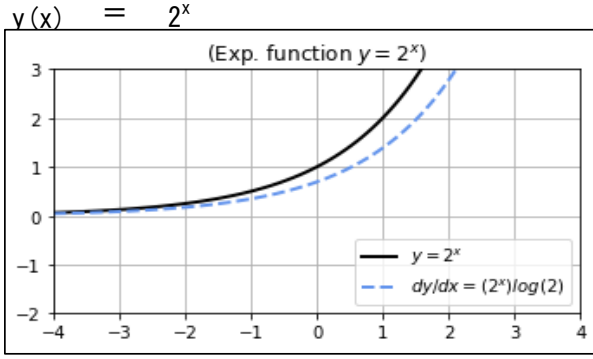
※1 「e（≒ 2.718281828459）」 は「ネイピア数（Napier's constant）」

(4) 指数関数とその導関数のグラフ例

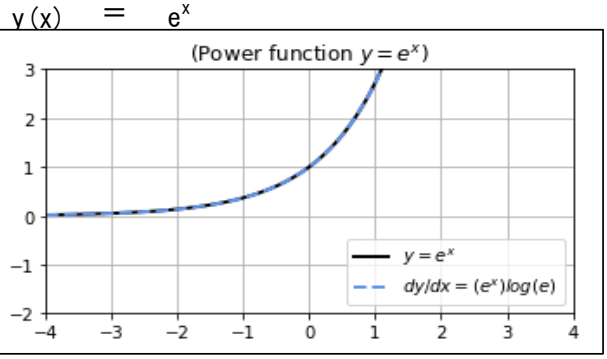
（「(5) 指数関数とその導関数の実装例」を参照）

（凡例） 黒実線： 関数
青点線： その導関数

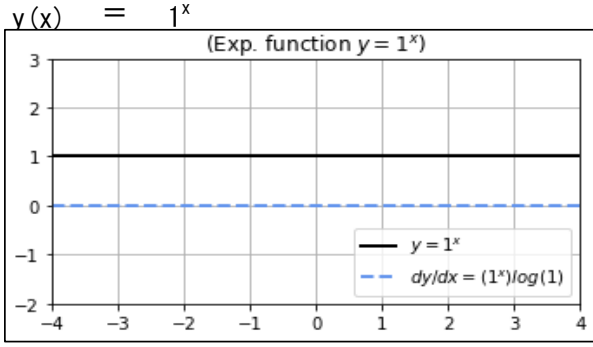
y(x) ＝ 2^x



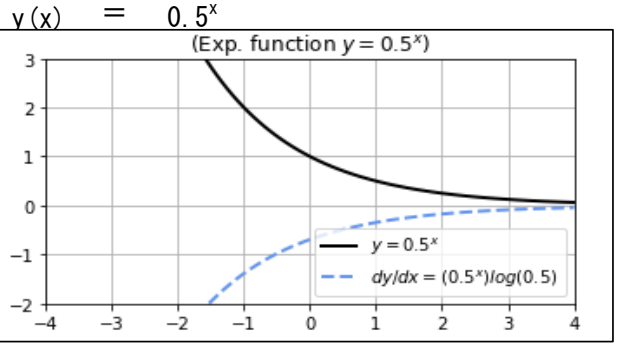
y(x) ＝ e^x



y(x) ＝ 1^x



y(x) ＝ 0.5^x



(5) 指数関数とその導関数の実装例

```
#**************************************************
# ﾘｽﾄ05-(03)-2_べき関数とその導関数の実装例
#**************************************************
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# 指数関数
def fExp(x, a):
    return a**x

# 指数関数の微分
def dfExp(x, a):
    return (a**x)*np.log(a)
```

```
# ネイピア数 e
e = np.exp(1)

# x データ列
x = np.linspace(-4, 4, 100)

# 指数関数とその導関数
y1 = fExp(x, 2)
y2 = fExp(x, e)
y3 = fExp(x, 1)
y4 = fExp(x, 0.5)
dy1 = dfExp(x, 2)
dy2 = dfExp(x, e)
dy3 = dfExp(x, 1)
dy4 = dfExp(x, 0.5)

# グラフ表示
plt.figure(figsize=(12,7))
plt.subplots_adjust(wspace=0.2, hspace=0.5)

# グラフ表示：指数関数 y=2^x
plt.subplot(2,2,1)
plt.title("(Exp. function $y=2^x$)")
plt.plot(x, y1, 'black', linewidth=2, label='$y=2^x$')
plt.plot(x, dy1, 'cornflowerblue', linewidth=2, label='$dy/dx=(2^x)log(2)$', linestyle='--')
plt.ylim(-2, 3)
plt.xlim(-4, 4)
plt.grid(True)
plt.legend(loc='lower right')

# グラフ表示：指数関数 y=e^x
plt.subplot(2,2,2)
plt.title("(Power function $y=e^x$)")
plt.plot(x, y2, 'black', linewidth=2, label='$y=e^x$')
plt.plot(x, dy2, 'cornflowerblue', linewidth=2, label='$dy/dx=(e^x)log(e)$', linestyle='--')
plt.ylim(-2, 3)
plt.xlim(-4, 4)
plt.grid(True)
plt.legend(loc='lower right')

# グラフ表示：指数関数 y=1^x
plt.subplot(2,2,3)
plt.title("(Exp. function $y=1^x$)")
plt.plot(x, y3, 'black', linewidth=2, label='$y=1^x$')
plt.plot(x, dy3, 'cornflowerblue', linewidth=2, label='$dy/dx=(1^x)log(1)$', linestyle='--')
plt.ylim(-2, 3)
plt.xlim(-4, 4)
plt.grid(True)
plt.legend(loc='lower right')

# グラフ表示：指数関数 y=0.5^x
plt.subplot(2,2,4)
plt.title("(Exp. function $y=0.5^x$)")
plt.plot(x, y4, 'black', linewidth=2, label='$y=0.5^x$')
plt.plot(x, dy4, 'cornflowerblue', linewidth=2, label='$dy/dx=(0.5^x)log(0.5)$', linestyle='--')
plt.ylim(-2, 3)
plt.xlim(-4, 4)
plt.grid(True)
plt.legend(loc='lower right')

plt.show()
```

(3.3) 対数関数

(1) 対数関数とは

対数関数

・「対数（ﾀｲｽｳ、logarithm)」とは、ある数 x を数 a のべき乗 a^y として表した場合の、べき指数 y のことを言います。

$$x = a^y$$

y は「底を a とする x の対数（base a logarithm of x)」と呼ばれ、$\log_a x$ と表現します。

$$y = \log_a x \quad (a>0,\ a\neq 1) \quad \cdots(式3.3\text{-}1)$$

・尚、x は「真数（ｼﾝｽｳ、antilogarithm)」と呼ばれます。
また、底 a は 1 でない正数であり（$a \neq 1,\ a > 0$)、
真数 x が正数である場合（$x > 0$）について定義されます。

・実数の対数関数 $\log_a x$ は底 a に対する指数関数 a^x の逆関数になっています。

（例）

・10 を底とする x の対数関数

$$y(x) = \log_{10} x$$

$$y(10^2) = \log_{10}(10^2) = 2$$
$$y(10^1) = \log_{10}(10^1) = 1$$
$$y(10^0) = \log_{10}(10^0) = 0$$
$$y(10^{-1}) = \log_{10}(10^{-1}) = -1$$
$$y(10^{-2}) = \log_{10}(10^{-2}) = -2$$

(2) 対数関数の性質

対数関数の性質を幾つか列挙しておきます

・1の対数
$$\log_a 1 = 0 \quad (a>0,\ a\neq 1) \quad \cdots(式3.3\text{-}2)$$
・底と同じ値の対数
$$\log_a a = 1 \quad (a>0,\ a\neq 1) \quad \cdots(式3.3\text{-}3)$$
・積の対数
$$\log_a(x \cdot y) = \log_a x + \log_a y \quad (a>0,\ a\neq 1) \quad \cdots(式3.3\text{-}4)$$
・商の対数
$$\log_a(x / y) = \log_a x - \log_a y \quad (a>0,\ a\neq 1) \quad \cdots(式3.3\text{-}5)$$
・べき乗の対数
$$\log_a x^n = n \cdot \log_a x \quad (a>0,\ a\neq 1) \quad \cdots(式3.3\text{-}6)$$
・底の変換
$$\log_a x = \log_b x \,/\, \log_b a \quad (a,b>0,\ a,b\neq 1) \quad \cdots(式3.3\text{-}7)$$

(3) 対数関数の導関数

対数関数の導関数

・aを底とする対数(base a logarithm)

$$\frac{d}{dx}\log_a(x) = 1/x\log(a) \qquad (a>0, a\neq 1) \qquad \cdots(式3.3\text{-}8)$$

・自然対数(natural logarithm) (※2)

$$\frac{d}{dx}\log(x) = 1/x \qquad \cdots(式3.3\text{-}9)$$

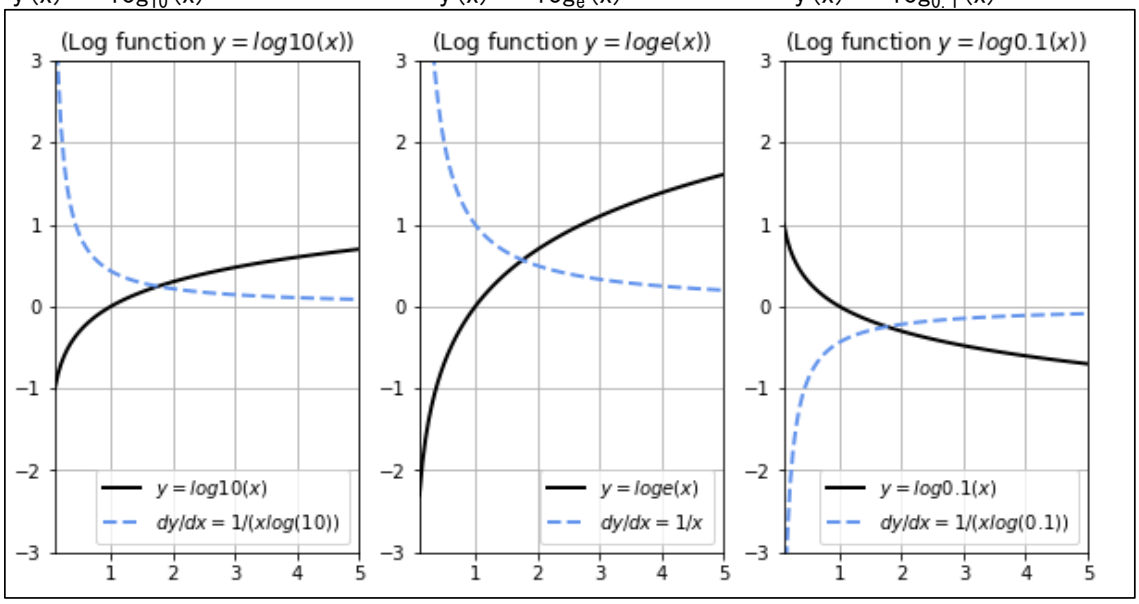
※2「自然対数 (ｼｾﾞﾝﾀｲｽｳ、natural logarithm)」は、
ネイピア数 e を底とする対数で、
「$\log_e(x)$」の代わりに、底 e を記述しない「log(x)」、
あるいは natural の頭文字 n から、「ln(x)」と記述することが多い。

(4) 対数関数とその導関数のグラフ例

(「(5) 対数関数とその導関数の実装例」を参照)

(凡例) 黒実線： 関数
青点線： その導関数

$y(x) = \log_{10}(x)$ $y(x) = \log_e(x)$ $y(x) = \log_{0.1}(x)$



【出典・参考】

⇒ https://ja.wikipedia.org/wiki/対数

⇒ https://dic.nicovideo.jp/a/対数

⇒ 「数学公式事典」 黒田孝朗・須田貞之著 文研出版 1978

(5) 対数関数とその導関数の実装例

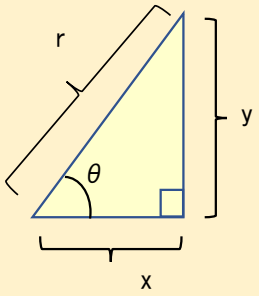
```
#************************************************
# ﾘｽﾄ05-(03)-3_対数関数とその導関数の実装例
#************************************************
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# 対数関数
def fLog(x, a):
    return  np.log(x) / np.log(a)

# 対数関数の微分
def dfLog(x, a):
    return 1 /( x * np.log(a) )

# ネイピア数 e
e = np.exp(1)

# x データ列
x = np.linspace(0.1, 5, 100)

# 対数関数とその導関数
y1 = fLog(x, 10)
y2 = fLog(x, e)
y3 = fLog(x, 0.1)
dy1 = dfLog(x, 10)
dy2 = dfLog(x, e)
dy3 = dfLog(x, 0.1)

# グラフ表示
plt.figure(figsize=(10,5))
plt.subplots_adjust(wspace=0.2, hspace=0.5)

# グラフ表示：対数関数 y=log10(x)
plt.subplot(1,3,1)
plt.title("(Log function $y=log10(x)$)")
plt.plot(x, y1, 'black', linewidth=2, label='$y=log10(x)$')
plt.plot(x, dy1, 'cornflowerblue', linewidth=2, label='$dy/dx=1/(x log(10))$', linestyle='--')
plt.ylim(-3, 3)
plt.xlim(0.1, 5)
plt.grid(True)
plt.legend(loc='lower right')

# グラフ表示：対数関数 y=loge(x)
plt.subplot(1,3,2)
plt.title("(Log function $y=loge(x)$)")
plt.plot(x, y2, 'black', linewidth=2, label='$y=loge(x)$')
plt.plot(x, dy2, 'cornflowerblue', linewidth=2, label='$dy/dx=1/x$', linestyle='--')
plt.ylim(-3, 3)
plt.xlim(0.1, 5)
plt.grid(True)
plt.legend(loc='lower right')

# グラフ表示：対数関数 y=log0.1(x)
plt.subplot(1,3,3)
plt.title("(Log function $y=log0.1(x)$)")
plt.plot(x, y3, 'black', linewidth=2, label='$y=log0.1(x)$')
plt.plot(x, dy3, 'cornflowerblue', linewidth=2, label='$dy/dx=1/(x log(0.1))$', linestyle='--')
plt.ylim(-3, 3)
plt.xlim(0.1, 5)
plt.grid(True)
plt.legend(loc='lower right')

plt.show()
```

(3.4) 三角関数

(1) 三角関数とは

三角関数

・「三角関数（ｻﾝｶｸｶﾝｽｳ、trigonometric function)」とは、
三角形の角の大きさと辺の長さの間の関係を記述する関数群を言います。
単位円を用いた定義に由来する呼び名として、
「円関数（ｴﾝｶﾝｽｳ、circular function)」
と呼ばれることがあります。



・以下の様な関数があります：

・正弦（sine）	$\sin(\theta)$	$= y / r$		…(式3.4-1)
・余弦（cosine）	$\cos(\theta)$	$= x / r$		…(式3.4-2)
・正接（tangent）	$\tan(\theta)$	$= y / x$	$(x \neq 0)$	…(式3.4-3)
・余割（cosecant）	$\mathrm{cosec}(\theta)$	$= 1 / \sin(\theta) = r / y$	$(y \neq 0)$	…(式3.4-4)
・正割（secant）	$\sec(\theta)$	$= 1 / \cos(\theta) = r / x$	$(x \neq 0)$	…(式3.4-5)
・余接（cotangent）	$\cot(\theta)$	$= 1 / \tan(\theta) = x / y$	$(y \neq 0)$	…(式3.4-6)

・三角形の角の大きさθの単位として、
「度（ﾄﾞ、degree、[°])」と「ラジアン（radian、[rad])」があります。
「度」と「ラジアン」の間には以下の様な関係があります。

π [rad] ＝ 180[°]　　（π：ﾊﾟｲ、円周率）　　…(式3.4-7)

∴ $\pi/2$[rad] ＝ 90[°]
2π [rad] ＝ 360[°]

（例）

$\sin(30°) = \sin(\pi/6) = 1/2$
$\cos(30°) = \cos(\pi/6) = \sqrt{3}/2$
$\sin(45°) = \sin(\pi/4) = 1/\sqrt{2}$
$\cos(45°) = \cos(\pi/4) = 1/\sqrt{2}$
$\sin(60°) = \sin(\pi/3) = \sqrt{3}/2$
$\cos(60°) = \cos(\pi/3) = 1/2$
$\sin(90°) = \sin(\pi/2) = 1$
$\cos(90°) = \cos(\pi/2) = 0$

(2) 三角関数の性質

三角関数の性質を幾つか列挙しておきます

・tanとcos,sinとの関係

$\tan(\theta) = \sin(\theta) / \cos(\theta)$ $(\theta \neq \pi/2)$ …(式3.4-8)

・二乗の和

$\sin^2(\theta) + \cos^2(\theta) = 1$ …(式3.4-9)

・$-\theta$の三角関数

$\sin(-\theta) = -\sin(\theta)$ …(式3.4-10)

$\cos(-\theta) = \cos(\theta)$ …(式3.4-11)

$\tan(-\theta) = -\tan(\theta)$ …(式3.4-12)

・$\pi/2 \pm\theta$の三角関数

$\sin(\pi/2 \pm\theta) = \cos(\theta)$ …(式3.4-13)

$\cos(\pi/2 \pm\theta) = \mp \sin(\theta)$ (複号同順) …(式3.4-14)

$\tan(\pi/2 \pm\theta) = \mp \cot(\theta)$ (複号同順) …(式3.4-15)

・$\pi \pm\theta$の三角関数

$\sin(\pi \pm\theta) = \mp \sin(\theta)$ (複号同順) …(式3.4-16)

$\cos(\pi \pm\theta) = -\cos(\theta)$ …(式3.4-17)

$\tan(\pi \pm\theta) = \pm\tan(\theta)$ (複号同順) …(式3.4-18)

・加法定理

$\sin(\alpha \pm\beta) = \sin(\alpha)\cos(\beta) \pm \cos(\alpha)\sin(\beta)$ (複号同順) …(式3.4-19)

$\cos(\alpha \pm\beta) = \cos(\alpha)\cos(\beta) \mp \sin(\alpha)\sin(\beta)$ (複号同順) …(式3.4-20)

$\tan(\alpha \pm\beta) = \{ \tan(\alpha) \pm \tan(\beta) \} / \{ 1 \mp \tan(\alpha)\tan(\beta) \}$ (複号同順) …(式3.4-21)

・周期性

$\sin(\theta) = \sin(\theta + (n * 2\pi))$ (n：任意整数) …(式3.4-22)

$\cos(\theta) = \cos(\theta + (n * 2\pi))$ (n：任意整数) …(式3.4-23)

$\tan(\theta) = \tan(\theta + (n * \pi))$ (n：任意整数) …(式3.4-24)

(3) 三角関数の導関数

三角関数の導関数

・正弦 (sine) $d \sin(x)/dx = \cos(x)$ …(式3.4-25)

・余弦 (cosine) $d \cos(x)/dx = -\sin(x)$ …(式3.4-26)

・正接 (tangent) $d \tan(x)/dx = \sec^2(x)$ …(式3.4-27)

・余割 (cosecant) $d \mathrm{cosec}(x)/dx = -\mathrm{cosec}(x)\cot(x)$ …(式3.4-28)

・正割 (secant) $d \sec(x)/dx = \sec(x)\tan(x)$ …(式3.4-29)

・余接 (cotangent) $d \cot(x)/dx = -\mathrm{cosec}^2(x)$ …(式3.4-30)

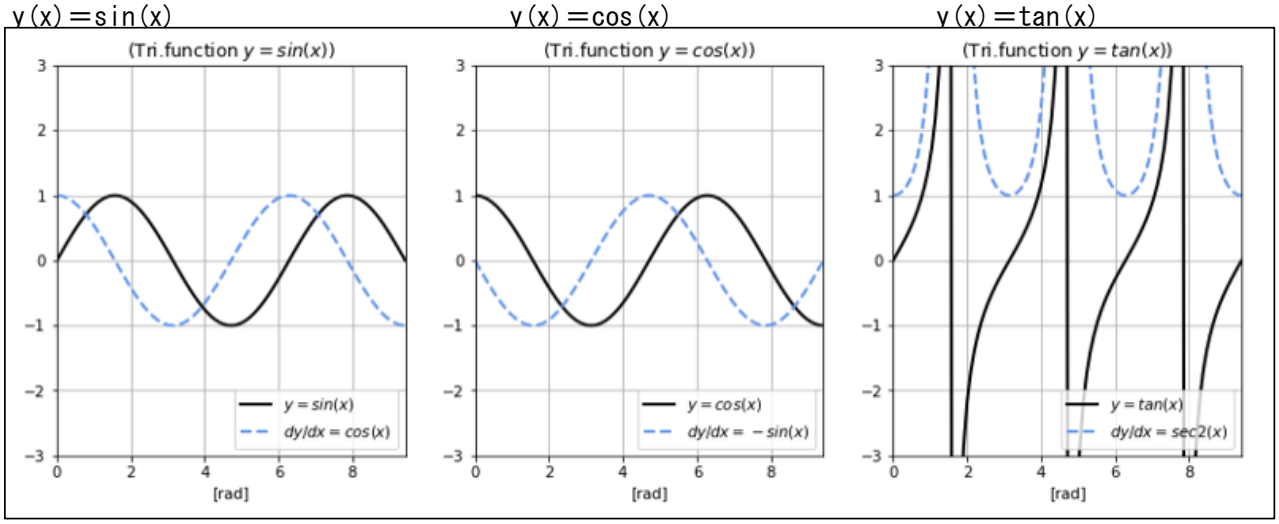
(4) 三角関数とその導関数のグラフ例

・三角関数は、グラフを見てもわかるように周期的な関数です。
sin, cos は 360°（2π[rad]）で、tan は 180°（π[rad]）で、
各々周期的に値が変動します。

（凡例） 黒実線： 関数
青点線： その導関数

（「(5) 三角関数とその導関数の実装例」参照）

y(x)＝sin(x)　　y(x)＝cos(x)　　y(x)＝tan(x)



(5) 三角関数とその導関数の実装例

```
#**************************************************
# ﾘｽﾄ05-(03)-4_三角関数とその導関数の実装例
#**************************************************
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# 三角関数 sin の微分
def dfSin(x):
    return np.cos(x)

# 三角関数 cos の微分
def dfCos(x):
    return  -1 * np.sin(x)

# 三角関数 tan の微分
def dfTan(x):
    return  1 / (np.cos(x))** 2

# x データ列 (0～3π [rad])
x = np.linspace(0, 3*np.pi, 100)

# 三角関数とその導関数
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)
dy1 = dfSin(x)
dy2 = dfCos(x)
dy3 = dfTan(x)

# グラフ表示
plt.figure(figsize=(14,5))
plt.subplots_adjust(wspace=0.2, hspace=0.5)

# グラフ表示：三角関数 y=sin(x)
```

```
plt.subplot(1,3,1)
plt.title("(Tri.function $y=sin(x)$)")
plt.plot(x, y1, 'black', linewidth=2, label='$y=sin(x)$')
plt.plot(x, dy1, 'cornflowerblue', linewidth=2, label='$dy/dx=cos(x)$', linestyle='--')
plt.ylim(-3, 3)
plt.xlim(0, 3*np.pi)
plt.xlabel("[rad]")
plt.grid(True)
plt.legend(loc='lower right')

# グラフ表示：三角関数 y=cos(x)
plt.subplot(1,3,2)
plt.title("(Tri.function $y=cos(x)$)")
plt.plot(x, y2, 'black', linewidth=2, label='$y=cos(x)$')
plt.plot(x, dy2, 'cornflowerblue', linewidth=2, label='$dy/dx=-sin(x)$', linestyle='--')
plt.ylim(-3, 3)
plt.xlim(0, 3*np.pi)
plt.xlabel("[rad]")
plt.grid(True)
plt.legend(loc='lower right')

# グラフ表示：三角関数 y=tan(x)
plt.subplot(1,3,3)
plt.title("(Tri.function $y=tan(x)$)")
plt.plot(x, y3, 'black', linewidth=2, label='$y=tan(x)$')
plt.plot(x, dy3, 'cornflowerblue', linewidth=2, label='$dy/dx=sec2(x)$', linestyle='--')
plt.ylim(-3, 3)
plt.xlim(0, 3*np.pi)
plt.xlabel("[rad]")
plt.grid(True)
plt.legend(loc='lower right')

plt.show()
```

【出典・参考】

⇒ https://ja.wikipedia.org/wiki/三角関数

⇒ https://ja.wikipedia.org/wiki/三角法

⇒ https://ja.wikipedia.org/wiki/ラジアン

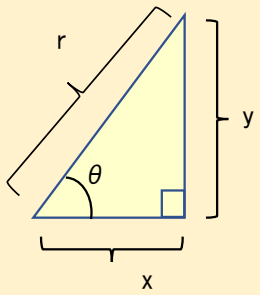
⇒ 「数学公式事典」 黒田孝朗・須田貞之著 文研出版 1978

(3.5) 逆三角関数

(1) 逆三角関数とは

逆三角関数

・「逆三角関数（ｷﾞｬｸｻﾝｶｸｶﾝｽｳ、inverse trigonometric function)」とは、
三角関数の逆関数を言います。
三角形の辺の長さから、辺同士の角度を求めるのに用いられます。



・以下の様な関数があります：

・逆正弦（arc sine）

…正弦関数 $z = \sin(\theta) = y/r$ の逆関数 $\theta = \arcsin(z) = \sin^{-1}(z)$ （$-1 \leq z \leq 1$、$-\pi/2 \leq \theta \leq \pi/2$） …(式3.5-1)

・逆余弦（arc cosine）

…余弦関数 $z = \cos(\theta) = x/r$ の逆関数 $\theta = \arccos(z) = \cos^{-1}(z)$ （$-1 \leq z \leq 1$、$0 \leq \theta \leq \pi$） …(式3.5-2)

・逆正接（arc tangent）

…正接関数 $z = \tan(\theta) = y/x$ の逆関数 $\theta = \arctan(z) = \tan^{-1}(z)$ （$-\infty \leq z \leq \infty$、$-\pi/2 \leq \theta \leq \pi/2$） …(式3.5-3)

・逆余接（arc cotangent）

…余接関数 $z = \cot(\theta) = x/y$ の逆関数 $\theta = \operatorname{arccot}(z) = \cot^{-1}(z)$ （$-\infty \leq z \leq \infty$、$0 \leq \theta \leq \pi$） …(式3.5-4)

・「三角関数の性質」でも見たように、三角関数は周期的な関数です。
sin, cos は 360°（2π[rad]）で、
tan は 180°（π[rad]）で、各々周期的に値が変動します。

$\sin(\theta) = \sin(\theta + (n * 2\pi))$ (n：任意整数)
$\cos(\theta) = \cos(\theta + (n * 2\pi))$ (n：任意整数)
$\tan(\theta) = \tan(\theta + (n * \pi))$ (n：任意整数)
$\cot(\theta) = \cot(\theta + (n * \pi))$ (n：任意整数)

また、「三角関数の性質」でも見たように、
一つの周期内でも同じ関数値を取る定義域があります。

$\sin(0) = \sin(\pi)$
$\cos(\pi/2) = \cos(3\pi/2)$ など

つまり、逆三角関数は多価関数となっています。
逆関数を定義するにあたって、
(式3.5-1)～(式3.5-4) のようにθの範囲を指定しておくことにより、
一価関数とします（定義域と値域を、１対１に対応させます）。
こうして得られる関数の値θを「主値（ｼｭﾁ, principal value)」と言います。

（例）

$\sin(30°)$	＝	$\sin(\pi/6)$	＝	$1/2$	、	$\arcsin(1/2)$	＝	$\pi/6$
$\cos(30°)$	＝	$\cos(\pi/6)$	＝	$\sqrt{3}/2$	、	$\arccos(\sqrt{3}/2)$	＝	$\pi/6$
$\sin(45°)$	＝	$\sin(\pi/4)$	＝	$1/\sqrt{2}$	、	$\arcsin(1/\sqrt{2})$	＝	$\pi/4$
$\cos(45°)$	＝	$\cos(\pi/4)$	＝	$1/\sqrt{2}$	、	$\arccos(1/\sqrt{2})$	＝	$\pi/4$
$\sin(60°)$	＝	$\sin(\pi/3)$	＝	$\sqrt{3}/2$	、	$\arcsin(\sqrt{3}/2)$	＝	$\pi/3$
$\cos(60°)$	＝	$\cos(\pi/3)$	＝	$1/2$	、	$\arccos(1/2)$	＝	$\pi/3$
$\sin(90°)$	＝	$\sin(\pi/2)$	＝	1	、	$\arcsin(1)$	＝	$\pi/2$
$\cos(90°)$	＝	$\cos(\pi/2)$	＝	0	、	$\arccos(0)$	＝	$\pi/2$

(2) 逆三角関数の導関数

逆三角関数の導関数は、「逆関数の微分」の公式で導くことが出来ます。

・逆正弦（arc sine）

$$\frac{d}{dz}\sin^{-1}(z) = 1/\sqrt{1-z^2} \quad (-1<x<1) \quad \cdots(式3.5\text{-}5)$$

・逆余弦（arc cosine）

$$\frac{d}{dz}\cos^{-1}(z) = -1/\sqrt{1-z^2} \quad (-1<x<1) \quad \cdots(式3.5\text{-}6)$$

・逆正接（arc tangent）

$$\frac{d}{dz}\tan^{-1}(z) = 1/(1+z^2) \quad \cdots(式3.5\text{-}7)$$

・逆余接（arc cotangent）

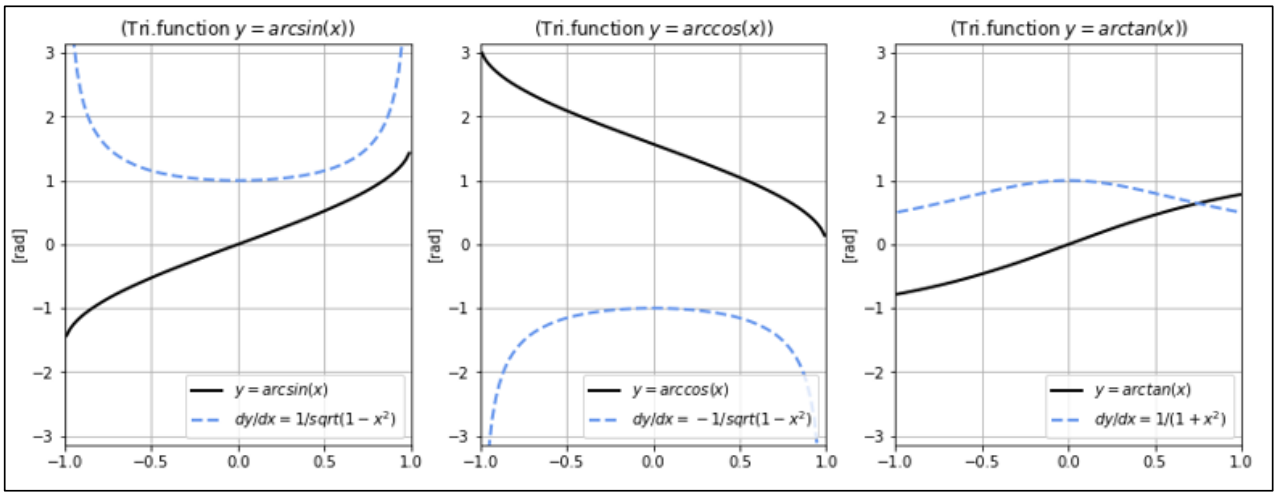
$$\frac{d}{dz}\cot^{-1}(z) = -1/(1+z^2) \quad \cdots(式3.5\text{-}8)$$

(3) 逆三角関数とその導関数のグラフ例

（「(4) 逆三角関数とその導関数の実装例」参照）

（凡例）　黒実線： 関数
青点線： その導関数

y(x)＝arcsin(x)　　y(x)＝arccos(x)　　y(x)＝arctan(x)



(4) 逆三角関数とその導関数の実装例

```
#************************************************
# ﾘｽﾄ05-(03)-5_逆三角関数とその導関数の実装例
#************************************************
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# 逆三角関数 arcsin の微分
def dfArcSin(x):
    return 1 / np.sqrt(1 - x**2)

# 逆三角関数 arccos の微分
def dfArcCos(x):
    return -1 / np.sqrt(1 - x**2)

# 逆三角関数 arctan の微分
def dfArcTan(x):
    return  1 / (1 + x**2)

# x データ列 (-1<x<1)
x = np.linspace(-0.99, 0.99, 100)

# 逆三角関数とその導関数
y1 = np.arcsin(x)
y2 = np.arccos(x)
y3 = np.arctan(x)
dy1 = dfArcSin(x)
dy2 = dfArcCos(x)
dy3 = dfArcTan(x)

# グラフ表示
plt.figure(figsize=(14,5))
plt.subplots_adjust(wspace=0.2, hspace=0.5)

# グラフ表示：逆三角関数 y=arcsin(x)
plt.subplot(1,3,1)
plt.title("(Tri.function $y=arcsin(x)$)")
plt.plot(x, y1, 'black', linewidth=2, label='$y=arcsin(x)$')
plt.plot(x, dy1, 'cornflowerblue', linewidth=2, label='$dy/dx=1/sqrt(1-x^2)$', linestyle='--')
plt.ylim(-np.pi, np.pi)
plt.xlim(-1, 1)
plt.ylabel("[rad]")
plt.grid(True)
plt.legend(loc='lower right')

# グラフ表示：逆三角関数 y=arccos(x)
plt.subplot(1,3,2)
plt.title("(Tri.function $y=arccos(x)$)")
plt.plot(x, y2, 'black', linewidth=2, label='$y=arccos(x)$')
plt.plot(x, dy2, 'cornflowerblue', linewidth=2, label='$dy/dx=-1/sqrt(1-x^2)$', linestyle='--')
plt.ylim(-np.pi, np.pi)
plt.xlim(-1, 1)
plt.ylabel("[rad]")
plt.grid(True)
plt.legend(loc='lower right')

# グラフ表示：逆三角関数 y=arctan(x)
plt.subplot(1,3,3)
plt.title("(Tri.function $y=arctan(x)$)")
plt.plot(x, y3, 'black', linewidth=2, label='$y=arctan(x)$')
plt.plot(x, dy3, 'cornflowerblue', linewidth=2, label='$dy/dx=1/(1+x^2)$', linestyle='--')
plt.ylim(-np.pi, np.pi)
plt.xlim(-1, 1)
plt.ylabel("[rad]")
```

```
plt.grid(True)
plt.legend(loc='lower right')

plt.show()
```

【出典・参考】

⇒ https://ja.wikipedia.org/wiki/逆三角関数

⇒ https://ja.wikipedia.org/wiki/主値

⇒ 「数学公式事典」 黒田孝朗・須田貞之著 文研出版 1978

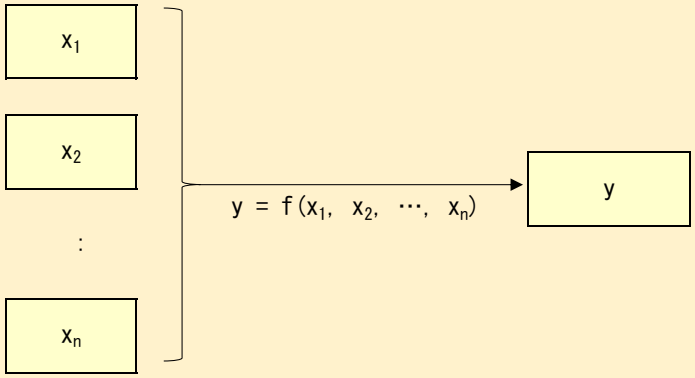
(4) 偏微分

・多変数関数について、着目変数について微分を施すのが偏微分です。
機械学習では、多くの重み変数についての偏微分を用いて、モデルの最適化をします。

(4.1) 多変数関数

・機械学習で出てくるのは、複数の変数を持つ多変数関数です。

多変数関数
・変数 y が複数の変数(x_1, x_2, …, x_n)の関数f となっているとき、
関数f を「多変数関数 (ﾀﾍﾝｽｳｶﾝｽｳ、multivariable function)」と言います。



（例）

サイズが（mピクセル×nピクセル）で、256階調のモノクロ画像は、

画像上の座標（x,y）に対する輝度 I(x,y)（0≦I(x,y)≦255、1≦x≦m、1≦y≦n）
という多変数関数になります。

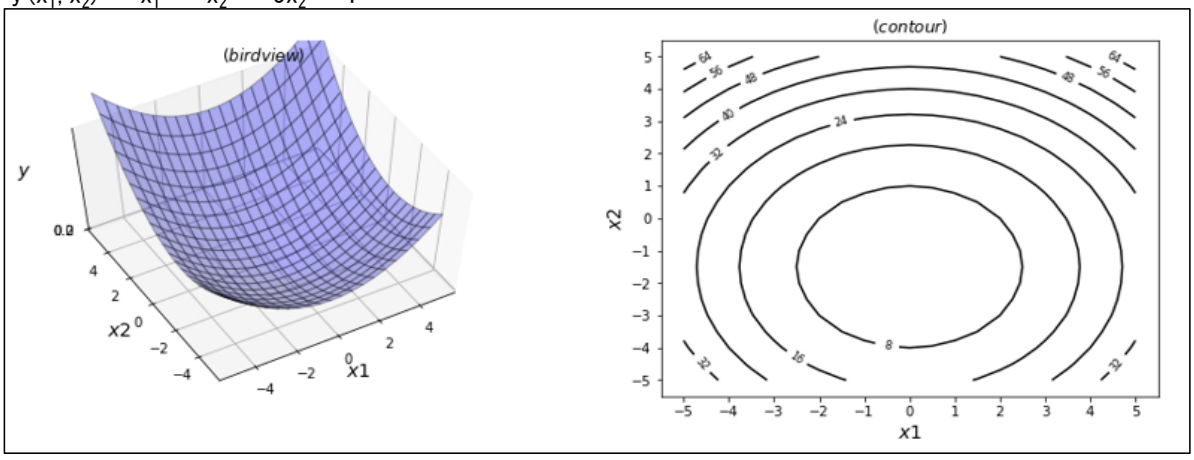
（例）

緯度x, 経度y における標高h は、

緯度・経度（x,y）に対する標高 h(x,y)（-90° ≦x≦+90° 、-180° ≦y≦+180° ）
という多変数関数になります。

・以下の例は多変数関数(yはx_1, x_2 の2変数の関数)のグラフです。
（「ﾘｽﾄ05-(04)-1_多変数関数」参照）

$y(x_1, x_2) = x_1^2 + x_2^2 + 3x_2 + 4$



（鳥瞰図）　　（等高線）

ﾘｽﾄ05-(04)-1_多変数関数

```
#**************************************************
# ﾘｽﾄ05-(04)-1_多変数関数
#**************************************************
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline

# 関数「y(x1,x2) = x1**2 + x2**2 + 3*x2 + 4」
def f(x1, x2):
    return x1**2 + x2**2 + 3*x2 + 4

# データの範囲とグリッド
x_range = 5
dx = 0.5
x1 = np.arange(-x_range, x_range+dx, dx)
x2 = np.arange(-x_range, x_range+dx, dx)
xn = x1.shape[0]
mesh1, mesh2 = np.meshgrid(x1, x2)

# 関数値を算出
ff = np.zeros( (len(x1), len(x2)) )
for i0 in range(xn):
    for i1 in range(xn):
        ff[i1, i0] = f( x1[i0], x2[i1] )

# グラフ描画を複数行う為の準備
plt.figure(figsize=(15, 5))
plt.subplots_adjust(wspace=0.3, hspace=0.3)

# 関数の面描画準備
ax = plt.subplot(1, 2, 1, projection='3d' )
ax.plot_surface(mesh1, mesh2, ff, rstride=1, cstride=1,
                alpha=0.3, color='blue', edgecolor='black')
ax.set_zticks(( 0, 0.2 ))
ax.view_init( 60, -120 )
ax.set_title('$(bird view)$')
ax.set_xlim(-x_range - 0.5, x_range + 0.5)
ax.set_ylim(-x_range - 0.5, x_range + 0.5)
ax.set_zlim(0, 50)
ax.set_xlabel('$x1$', fontsize=14)
ax.set_ylabel('$x2$', fontsize=14)
ax.set_zlabel('$y$', fontsize=14)

# 関数の等高線描画準備
plt.subplot(1, 2, 2)
cont = plt.contour(mesh1, mesh2, ff, 10, colors='k')
cont.clabel(fmt='%2.0f', fontsize=8)
plt.xticks(range(-x_range, x_range + 1, 1))
plt.yticks(range(-x_range, x_range + 1, 1))
plt.xlim(-x_range - 0.5, x_range + 0.5)
plt.ylim(-x_range - 0.5, x_range + 0.5)
plt.xlabel('$x1$', fontsize=14)
plt.ylabel('$x2$', fontsize=14)
plt.title('$(contour)$')

# 表示
plt.show()
```

【出典・参考】

多変数関数⇒ http://www.ftext.org/text/subsubsection/2382

(4.2) 偏微分可能・偏導関数・偏微分

・多変数関数について、着目変数に絞って微分を施すのが偏微分です。

偏微分可能・偏導関数・偏微分

・n個の変数$(x_1,\ x_2,\ \cdots,\ x_n)$の多変数関数 $y = f(x_1,\ x_2,\ \cdots,\ x_n)$ において、
一つの変数 x_k $(k=1,2,\cdots,n)$ 以外を固定すれば、

$$y = f(x_1,\ x_2,\ \cdots,\ x_n)$$

は、一つの変数 x_k だけの関数となります(※1)。
この変数 x_k の関数の導関数

$$\lim_{h\to 0} \left(f(x_1, x_2, \cdots, x_k+h, \cdots, x_n) - f(x_1, x_2, \cdots, x_k, \cdots, x_n) \right) \ / \ h$$

が存在する時、
関数 $y = f(x_1,\ x_2,\ \cdots,\ x_n)$ は、x_k で「偏微分可能 (ﾍﾝﾋﾞﾌﾞﾝｶﾉｳ、partial differentiable)」
であるといい、その極限値としてできた $(x_1,\ x_2,\ \cdots,\ x_n)$ の新しい関数を
関数 $y = f(x_1,\ x_2,\ \cdots,\ x_n)$ のx_k に関する
「偏導関数 (ﾍﾝﾄﾞｳｶﾝｽｳ、partial derived function)」と呼び、以下の様に表現します。

$$f_{xk}$$
$$f_{xk}(x_1, x_2, \cdots, x_n)$$
$$\partial f(x_1, x_2, \cdots, x_n) \ / \ \partial x_k$$ （∂は「ﾃﾞﾙ」と呼びます）

・また、多変数関数 $y = f(x_1,\ x_2,\ \cdots,\ x_n)$ の x_k に関する偏導関数を求めることを、
関数 $y = f(x_1,\ x_2,\ \cdots,\ x_n)$ を x_k で「偏微分 (ﾍﾝﾋﾞﾌﾞﾝ、partial differentiate)」する
と言います。

(※1 多変数関数 $y = f(x_1,\ x_2,\ \cdots,\ x_n)$ は、
n個の座標軸$(x_1,\ x_2,\ \cdots,\ x_n)$上で定義された値からなる曲面であると捉えた時、
「一つの変数 x_k $(k=1,2,\cdots,n)$ 以外を固定」した場合、
この曲面を、変数 x_k軸と y軸が作る平面と平行な断面で見ていることになります。)

(4.3) 偏微分についての公式

・関数 $f_i(x_1, x_2, \cdots, x_n)$ $(i=1, 2, \cdots, N)$ は、x_k で偏微分可能な N個の関数とするとき、以下の公式があります。

(1) 関数の和・差・商・積の偏微分

和・差・商・積の偏微分

・和・差の偏微分

$$\partial \{ f_1(x_1, x_2, \cdots, x_n) \pm f_2(x_1, x_2, \cdots, x_n) \} / \partial x_k \quad \cdots(式4.3\text{-}1)$$
$$= \partial f_1(x_1, x_2, \cdots, x_n) / \partial x_k \pm \partial f_2(x_1, x_2, \cdots, x_n) / \partial x_k$$

・ｽｶﾗｰ λ 倍の偏微分

$$\partial \lambda f_i(x_1, x_2, \cdots, x_n) / \partial x_k \quad \cdots(式4.3\text{-}2)$$
$$= \lambda \partial f_i(x_1, x_2, \cdots, x_n) / \partial x_k$$

・総和の偏微分(和の拡張)

$$\partial \{ \sum_{i=1}^{N} f_i(x_1, x_2, \cdots, x_n) \} / \partial x_k \quad \cdots(式4.3\text{-}3)$$
$$= \sum_{i=1}^{N} \partial f_i(x_1, x_2, \cdots, x_n) / \partial x_k$$

（例）

・和・差・ｽｶﾗｰ倍の偏微分

$z(x, y) = ax^2 - by^2 + cxy + d$

$$\partial z / \partial x = \partial (ax^2 - by^2 + cxy + d) / \partial x$$
$$= \partial (ax^2) / \partial x + \partial (-by^2) / \partial x + \partial cxy / \partial x + \partial d / \partial x$$
$$= 2ax + 0 + cy + 0$$
$$= 2ax + cy$$

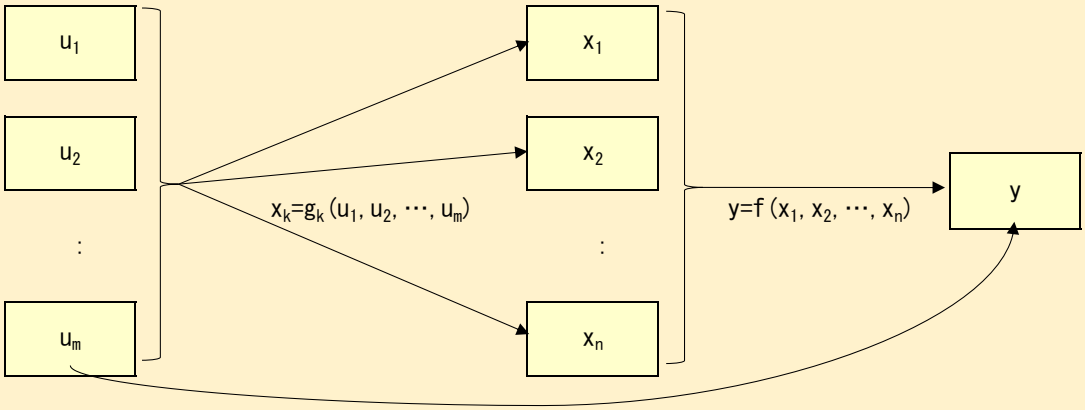
$$\partial z / \partial y = \partial (ax^2 - by^2 + cxy + d) / \partial y$$
$$= \partial (ax^2) / \partial y + \partial (-by^2) / \partial y + \partial cxy / \partial y + \partial d / \partial y$$
$$= 0 - 2by + cx + 0$$
$$= - 2by + cx$$

(2) 合成関数の偏微分と連鎖律

・合成関数の偏微分と連鎖律は、機械学習において誤差の逆伝搬の計算で用いられます。

・合成関数の偏微分と連鎖律

n個の変数$(x_1, x_2, \cdots, x_n)$の多変数関数 $y=f(x_1, x_2, \cdots, x_n)$ において、
変数 x_k $(k=1, 2, \cdots, n)$ が、m個の変数$(u_1, u_2, \cdots, u_m)$の多変数関数 $x_k=g_k(u_1, u_2, \cdots, u_m)$
となっているとします。



$$y = f(x_1, x_2, \cdots, x_n)$$
$$= f(g_1(u_1, u_2, \cdots, u_m),\ g_2(u_1, u_2, \cdots, u_m),\ \cdots,\ g_n(u_1, u_2, \cdots, u_m))$$

この時、n個の変数$(x_1, x_2, \cdots, x_n)$の多変数関数 $y=f(x_1, x_2, \cdots, x_n)$ は、
変数$(u_1, u_2, \cdots, u_m)$ の関数として、次式により与えられます：

$$y = f(x_1, x_2, \cdots, x_n)$$
$$= f(g_1(u_1, u_2, \cdots, u_m),\ g_2(u_1, u_2, \cdots, u_m),\ \cdots,\ g_n(u_1, u_2, \cdots, u_m)) \quad \cdots(式4.3\text{-}4)$$

この偏導関数に関して、
関数 $y=f(x_1, x_2, \cdots, x_n)$ の x_k $(k=1, 2, \cdots, n)$ に関する偏導関数 $\partial f/\partial x_k$ が連続で、
関数 $x_k=g_k(u_1, u_2, \cdots, u_m)$ が u_j $(j=1, 2, \cdots, m)$ について偏微分可能ならば、
その偏導関数は、次式により与えられます：

$$\partial y/\partial u_j = \partial f/\partial x_1 \cdot \partial x_1/\partial u_j + \partial f/\partial x_2 \cdot \partial x_2/\partial u_j + \cdots + \partial f/\partial x_n \cdot \partial x_n/\partial u_j \quad (j=1, 2, \cdots, m) \quad \cdots(式4.3\text{-}5)$$

または、

$$\partial y/\partial u_j = \sum_{k=1}^{n} \partial f/\partial x_k \cdot \partial x_k/\partial u_j \quad (j=1, 2, \cdots, m) \quad \cdots(式4.3\text{-}6)$$

または、

$$\partial y/\partial u_j = \sum_{k=1}^{n} \partial f/\partial g_k \cdot \partial g_k/\partial u_j \quad (j=1, 2, \cdots, m) \quad \cdots(式4.3\text{-}7)$$

これが合成関数の偏微分についての「連鎖律（れんさりつ、chain rule)」です。

（例）

a, b, c が定数で、$z(x,y)=(ax+by+c)^3$ の偏微分を合成関数の偏微分法により求めてみます。

$z=f(g(x,y))=g(x,y)^3$, $g(x) = ax+by+c$ として、

$$\partial f/\partial x \quad =_{※1} \quad \partial f/\partial g \cdot \partial g/\partial x$$
$$=_{※3} \quad (3g^2)a$$
$$= \quad 3a(ax+by+c)^2$$

※1：連鎖律
※2：$\partial f/\partial g=3g^2$、$\partial g/\partial y=b$
※3：$\partial f/\partial g=3g^2$、$\partial g/\partial x=a$

$$\partial f/\partial y \quad =_{※1} \quad \partial f/\partial g \cdot \partial g/\partial y$$
$$=_{※2} \quad (3g^2)b$$
$$= \quad 3b(ax+by+c)^2$$

【出典・参考】
導関数⇒ 「数学公式事典」 黒田孝朗・須田貞之著 文研出版 1978

(5) 勾配

・「勾配 (ｺｳﾊﾞｲ, gradient)」は、傾き具合のことですが、機械学習では、
ニューラルネットワークモデルの誤差関数の値を最小化する為に重みを調整する際に使用します。

(5.1) 多変数関数の近似公式

(1) 1変数関数の近似公式

・変数x の関数 $y = f(x)$ について、
変数x が x から $x+\Delta x$ まで変化する時、関数f の値 $y = f(x+\Delta x)$の値の近似値は、
導関数の定義(式2.1-2)を元に、次式で与えられます。

1変数関数の近似公式

$$f(x+\Delta x) \fallingdotseq f(x) + df(x)/dx \cdot \Delta x \qquad (\Delta x\text{は小さな数}) \qquad \cdots(\text{式5.1-1})$$

（公式の導出）

$$f'(x) = \lim_{\Delta x \to 0} \left(f(x+\Delta x) - f(x) \right) / \Delta x \qquad \cdots(\text{式2.1-2})\text{再掲}$$

$$\therefore \quad f'(x) \fallingdotseq \left(f(x+\Delta x) - f(x) \right) / \Delta x$$

$$\therefore \quad df(x)/dx \fallingdotseq \left(f(x+\Delta x) - f(x) \right) / \Delta x \qquad \cdots(\text{式2.1-3})\text{再掲}$$

$$\therefore \quad f(x+\Delta x) \fallingdotseq f(x) + df(x)/dx \cdot \Delta x \qquad (\Delta x\text{は小さな数}) \qquad \cdots(\text{式5.1-1})$$

・これは点(x) での関数値 f(x) に、変数x の変化率(df(x)/dx)×変化量(Δx)を加えたものになっています。

(2) 多変数関数の近似公式

・1変数関数の近似公式を多変数関数に拡張します。

・n個の変数(x_1, x_2, …, x_n) の多変数関数 $y = f(x_1, x_2, \cdots, x_n)$ において、
変数(x_1, x_2, …, x_n) から少し変化させた($x_1+\Delta x_1$, $x_2+\Delta x_2$, …, $x_n+\Delta x_n$)での
多変数関数 $y = f(x_1+\Delta x_1, x_2+\Delta x_2, \cdots, x_n+\Delta x_n)$の近似値は、
1変数関数の近似公式を拡張した、次式で与えられます。

多変数関数の近似公式

$$\begin{aligned} & f(x_1+\Delta x_1, x_2+\Delta x_2, \cdots, x_n+\Delta x_n) && \cdots(\text{式5.1-2}) \\ & \fallingdotseq f(x_1, x_2, \cdots, x_n) \\ & + \partial f(x_1, x_2, \cdots, x_n) / \partial x_1 \cdot \Delta x_1 && (\Delta x_1\text{は小さな数}) \\ & + \partial f(x_1, x_2, \cdots, x_n) / \partial x_2 \cdot \Delta x_2 && (\Delta x_2\text{は小さな数}) \\ & + \cdots \\ & + \partial f(x_1, x_2, \cdots, x_n) / \partial x_n \cdot \Delta x_n && (\Delta x_n\text{は小さな数}) \end{aligned}$$

・これは点(x_1, x_2, …, x_n) での関数値 $f(x_1, x_2, \cdots, x_n)$ に、
各変数x_k(k=1～n)毎の（変化率×変化量）の総和を加えたものになっています。

・これは「テイラー展開 (Taylor's expansion)」において、
Δx_k(k=1～n)の一次の項までを取り上げたものに相当します。

【出典・参考】

⇒「ﾃﾞｨｰﾌﾟﾗｰﾆﾝｸﾞがわかる数学入門」涌井良幸・貞美著 技術評論社 2017.04

(5.2) 多変数関数の変化量と勾配

・多変数関数の近似公式(式5.1-2)から、
n個の変数(x_1, x_2, …, x_n) の多変数関数 $y = f(x_1, x_2, \cdots, x_n)$ において、
変数(x_1, x_2, …, x_n) から少し変化させた($x_1+\Delta x_1$, $x_2+\Delta x_2$, …, $x_n+\Delta x_n$)での
多変数関数 $y = f(x_1, x_2, \cdots, x_n)$の変化量$\Delta y$ は、次式で与えられます。

関数 $y = f(x_1, x_2, \cdots, x_n)$ の変化量

$$\begin{aligned}\Delta y &= f(x_1+\Delta x_1, x_2+\Delta x_2, \cdots, x_n+\Delta x_n) - f(x_1, x_2, \cdots, x_n) \\ &\fallingdotseq \partial f(x_1, x_2, \cdots, x_n)/\partial x_1 \cdot \Delta x_1 \quad (\Delta x_1\text{は小さな数}) \\ &+ \partial f(x_1, x_2, \cdots, x_n)/\partial x_2 \cdot \Delta x_2 \quad (\Delta x_2\text{は小さな数}) \\ &+ \cdots \\ &+ \partial f(x_1, x_2, \cdots, x_n)/\partial x_n \cdot \Delta x_n \quad (\Delta x_n\text{は小さな数})\end{aligned}$$ …(式5.2-1)

・これをベクトルとその内積で表現することが出来ます。

・はじめに、n個の変数(x_1, x_2, …, x_n) の多変数関数 $y = f(x_1, x_2, \cdots, x_n)$ の、
各成分ごとの偏微分を並べた「勾配 (ｺｳﾊﾞｲ、ｸﾞﾗﾃﾞｨｴﾝﾄ、gradient)」ベクトルというベクトルを
導入します。

関数 $f(x_1, x_2, \cdots, x_n)$ の勾配ベクトル

$$\nabla f = (\partial f/\partial x_1, \partial f/\partial x_2, \cdots, \partial f/\partial x_n)$$ …(式5.2-2)

・次に、点(x_1, x_2, …, x_n) から変化させた点($x_1+\Delta x_1$, $x_2+\Delta x_2$, …, $x_n+\Delta x_n$)への
移動量(Δx_1, Δx_2, …, Δx_n)を、移動量ベクトル$\boldsymbol{\Delta x}$で表現します。

移動量ベクトル

$$\boldsymbol{\Delta x} = (\Delta x_1, \Delta x_2, \cdots, \Delta x_n)$$ …(式5.2-3)

・(式5.2-1)で表現された多変数関数 y = f(x1, x2, …, xn)の変化量Δy は、
これらのベクトルの内積になります。

関数 $f(x_1, x_2, \cdots, x_n)$ の変化量Δyのベクトル表現

$$\Delta y = \nabla f \cdot \boldsymbol{\Delta x}$$ …(式5.2-4)

∇f は勾配ベクトル $(\partial f/\partial x_1, \partial f/\partial x_2, \cdots, \partial f/\partial x_n)$
$\boldsymbol{\Delta x}$ は移動量ベクトル $(\Delta x_1, \Delta x_2, \cdots, \Delta x_n)$

【出典・参考】
⇒「ﾃﾞｨｰﾌﾟﾗｰﾆﾝｸﾞがわかる数学入門」涌井良幸・貞美著 技術評論社 2017.04

(5.3) 勾配降下法

・(式5.2-4)にあるように、
n個の変数(x_1, x_2, …, x_n) の多変数関数 $y = f(x_1, x_2, \cdots, x_n)$ について、
点(x_1, x_2, …, x_n) での関数 f の値と、
点(x_1, x_2, …, x_n) から移動量ベクトル(Δx_1, Δx_2, …, Δx_n)だけ変化させた
点($x_1+\Delta x_1$, $x_2+\Delta x_2$, …, $x_n+\Delta x_n$) での関数 f の値との変化量Δy は、
勾配ベクトル∇f と移動量ベクトルΔx の内積になります。

関数 $f(x_1, x_2, \cdots, x_n)$ の変化量Δy

$$\Delta y = \nabla f \cdot \Delta \mathbf{x} \qquad \cdots(式5.2\text{-}4)再掲$$

∇f は勾配ベクトル $(\partial f/\partial x_1, \partial f/\partial x_2, \cdots, \partial f/\partial x_n)$
$\Delta \mathbf{x}$ は移動量ベクトル $(\Delta x_1, \Delta x_2, \cdots, \Delta x_n)$

・この関係式から、
関数 $f(x_1, x_2, \cdots, x_n)$の変化量Δy を最小にするのは（つまり、関数f の値を最も減少させるのは）、
勾配ベクトル∇f と移動量ベクトルΔx の内積を最小にする、ということと同じになります。

内積が最小になるのは、勾配ベクトル∇f と移動量ベクトルΔx とが反対向きの場合です。
従って、点(x_1, x_2, …, x_n) から移動量ベクトル$\Delta \mathbf{x}$＝(Δx_1, Δx_2, …, Δx_n)だけ変化させた
点($x_1+\Delta x_1$, $x_2+\Delta x_2$, …, $x_n+\Delta x_n$)へ移動する時、
関数 $f(x_1, x_2, \cdots, x_n)$ の値が最も減少するのは、次の関係が満たされる時になります。

関数 $f(x_1, x_2, \cdots, x_n)$ の値を最も小さくする移動量ベクトル$\Delta \mathbf{x}$

$$\Delta \mathbf{x} = -\eta \nabla f \qquad \cdots(式5.3\text{-}1)$$

∇f は勾配ベクトル $(\partial f/\partial x_1, \partial f/\partial x_2, \cdots, \partial f/\partial x_n)$
$\Delta \mathbf{x}$ は移動量ベクトル $(\Delta x_1, \Delta x_2, \cdots, \Delta x_n)$
η は「学習係数、または、学習率 (ｶﾞｸｼｭｳｹｲｽｳ, ｶﾞｸｼｭｳﾘﾂ, learning rate)」
と呼ばれる正の小さな定数

この移動量ベクトル$\Delta \mathbf{x}$を採用することにより、
関数 $f(x_1, x_2, \cdots, x_n)$ を最速に減少させることが出来ます。
この手法は「勾配降下法 (ｺｳﾊﾞｲｺｳｶﾎｳ, gradient descent method)」と呼ばれるもので、
機械学習では、モデルの誤差関数を減少させて最適化するために、この手法を用います。

以下にその概要を記します。

・「勾配降下法 (ｺｳﾊﾞｲｺｳｶﾎｳ, gradient descent method)」あるいは、
「最急降下法 (ｻｲｷｭｳｺｳｶﾎｳ, steepest descent method)」と呼ばれる方法では、
以下の様な手順を踏んで、関数 $f(x_1, x_2, \cdots, x_n)$ を減少させます。

勾配降下法の基本的アルゴリズム

(1) iを繰り返しの通し番号として、i=1 から開始する。
(2) 学習係数 η (>0) を決定する(※1)。
(3) 初期位置 $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{in})$ を決定する。

(4) 以下の様な繰り返し終了判定を行う。
  (4.1) 点$\mathbf{x}_i$ での関数 $f(\mathbf{x}_i)$ の値が十分小さければ、本処理を終了する。
    この時は 点$\mathbf{x}_i$ が関数 f の最小値を与える点と判断する。
  (4.2) 上記(4.1)以外で、繰り返し回数が制限を超過した場合、本処理を終了する。
    この時は 最小値を与える点は見つかっておらず、モデル等の見直しを行う必要あり。
  (4.3) 上記(4.1)、(4.2)以外の場合、処理(5)へ。

(5) 点$\mathbf{x}_i$ からの移動量ベクトル$\Delta\mathbf{x}$ を(式5.3-1)により計算して点$\mathbf{x}_{i+1}$ へ移動する。
  $\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}$ …(式5.3-2)
  ここで、$\Delta\mathbf{x} = -\eta\left(\partial f(\mathbf{x}_i)/\partial x_1, \partial f(\mathbf{x}_i)/\partial x_2, \cdots, \partial f(\mathbf{x}_i)/\partial x_n\right)$ …(式5.3-1')

(6) i ← i+1 として手順(4)へ

(※1) 学習係数 η を、手順(4)～(6)の過程で動的に決定する方法もあります。

・尚、勾配法では、極小値が最小値とは限らないのに、最小値でない極小値にはまってしまうと
抜け出られない、という欠点があります。
これを「局所解問題 (ｷｮｸｼｮｶｲﾓﾝﾀﾞｲ、local minumum problem)」と言います。
これの解決手法として「確率的勾配法」がありますが、後の回で改めて言及します。

（例）

・以下の図は多変数関数 y と、その勾配降下法による最小値探索の例です。
（「ﾘｽﾄ05-(05)-1_多変数関数と勾配」参照）

$$y(x_1, x_2) \quad = \quad x_1^2 + x_2^2 + 3x_2 + 4$$

この関数が構成する面の勾配ベクトル∇y は、

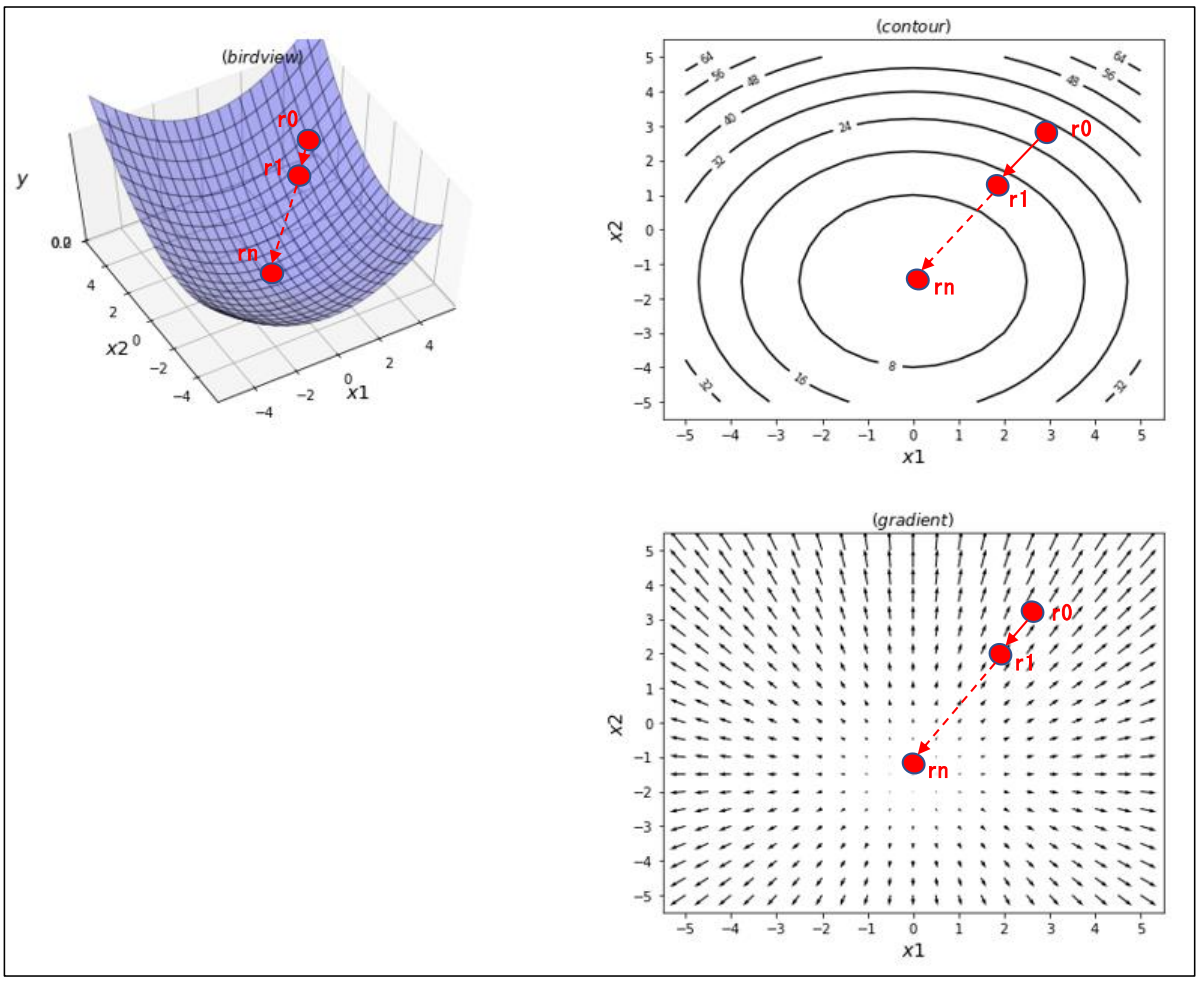
$$\nabla y \quad = \quad (\partial y/\partial x_1,\ \partial y/\partial x_2) \quad = \quad (2x_1,\ 2x_2+3)$$

で与えられ、それを示したものは右下図（gradient）です。

下図では、丸点とそれをつなぐ矢印は、初期位置 r_0 から始まって、

$$r_{i+1} \quad = \quad r_i - \eta(\partial y/\partial x_1,\ \partial y/\partial x_2) \qquad \eta\ (>0)：学習係数$$

という移動を繰り返して、
関数値が最小の点 r_n に落ち着く様子を示しています。



ﾘｽﾄ05-(05)-1_多変数関数と勾配

```
#****************************************************
# ﾘｽﾄ05-(05)-1_多変数関数と勾配
#****************************************************
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline

# 関数「y(x1,x2) = x1**2 + x2**2 + 3*x2 + 4」とその偏微分
def f(x1, x2):
    return x1**2 + x2**2 + 3*x2 + 4

def df_dx1(x1, x2):
```

```
    return 2*x1

def df_dx2(x1, x2):
    return 2*x2 + 3

# データの範囲とグリッド
x_range = 5
dx = 0.5
x1 = np.arange(-x_range, x_range+dx, dx)
x2 = np.arange(-x_range, x_range+dx, dx)
xn = x1.shape[0]
mesh1, mesh2 = np.meshgrid(x1, x2)

# 関数値と勾配を算出
ff = np.zeros( (len(x1), len(x2)) )
dff_dx1 = np.zeros( (len(x1), len(x2)) )
dff_dx2 = np.zeros( (len(x1), len(x2)) )
for i0 in range(xn):
    for i1 in range(xn):
        ff[i1, i0] = f( x1[i0], x2[i1] )
        dff_dx1[i1, i0] = df_dx1( x1[i0], x2[i1] )
        dff_dx2[i1, i0] = df_dx2( x1[i0], x2[i1] )

# グラフ描画を複数行う為の準備
plt.figure(figsize=(15, 12))
plt.subplots_adjust(wspace=0.2, hspace=0.3)

# 関数の面描画準備
ax = plt.subplot(2, 2, 1, projection='3d' )
ax.plot_surface(mesh1, mesh2, ff, rstride=1, cstride=1,
                alpha=0.3, color='blue', edgecolor='black')
ax.set_zticks(( 0, 0.2 ))
ax.view_init( 60, -120 )
ax.set_title('$(bird view)$')
ax.set_xlim(-x_range - 0.5, x_range + 0.5)
ax.set_ylim(-x_range - 0.5, x_range + 0.5)
ax.set_zlim(0, 50)
ax.set_xlabel('$x1$', fontsize=14)
ax.set_ylabel('$x2$', fontsize=14)
ax.set_zlabel('$y$', fontsize=14)

# 関数の等高線描画準備
plt.subplot(2, 2, 2)
cont = plt.contour(mesh1, mesh2, ff, 10, colors='k')
cont.clabel(fmt='%2.0f', fontsize=8)
plt.xticks(range(-x_range, x_range + 1, 1))
plt.yticks(range(-x_range, x_range + 1, 1))
plt.xlim(-x_range - 0.5, x_range + 0.5)
plt.ylim(-x_range - 0.5, x_range + 0.5)
plt.xlabel('$x1$', fontsize=14)
plt.ylabel('$x2$', fontsize=14)
plt.title('$(contour)$')

# 関数の勾配のベクトル描画準備
plt.subplot(2, 2, 4)
plt.quiver(mesh1, mesh2, dff_dx1, dff_dx2)
plt.xticks(range(-x_range, x_range + 1, 1))
plt.yticks(range(-x_range, x_range + 1, 1))
plt.xlim(-x_range - 0.5, x_range + 0.5)
plt.ylim(-x_range - 0.5, x_range + 0.5)
plt.xlabel('$x1$', fontsize=14)
plt.ylabel('$x2$', fontsize=14)
plt.title('$(gradient)$')

# 描画
plt.show()
```

(5.4) 勾配降下法と誤差逆伝搬法

・「順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸ (Feed forward neural networks、ﾌｨｰﾄﾞﾌｫﾜｰﾄﾞﾆｭｰﾗﾙﾈｯﾄﾜｰｸ)」の学習方法として、
「誤差逆伝搬法 (ｺﾞｻｷﾞｬｸﾃﾞﾝﾊﾟﾎｳ、backpropagation、ﾊﾞｯｸﾌﾟﾛﾊﾟｹﾞｰｼｮﾝ)」があります。

この誤差逆伝搬法では、ﾆｭｰﾗﾙﾈｯﾄﾜｰｸの出力で生じる誤差（教師信号との差）の情報を使って、
出力層の重みから中間層の重みへと入力方向と逆向きに重みを更新していきます。
誤差逆伝搬法は、1986年に「backwards propagation of errors (後方への誤差伝播)」の略から
ﾃﾞﾋﾞｯﾄﾞ･ﾗﾒﾙﾊｰﾄらによって命名されたものです。

この誤差逆伝搬法は、勾配降下法を順伝播型ﾆｭｰﾗﾙﾈｯﾄﾜｰｸの学習方法に適用したものです。
尚、誤差逆伝搬法では、偏微分を用いることからも分かるように、
各ﾆｭｰﾛﾝで使われる活性化関数が、微分可能である必要があります。

・入力データ全てに対する誤差関数の勾配を、更新の１ステップごとに計算する場合、
データが大きいとその計算にとても時間がかかってしまいます。
そのような場合には、データの一部で誤差関数の勾配を計算する、
「確率的勾配法 (ｶｸﾘﾂﾃｷｺｳﾊﾞｲｺｳｶﾎｳ、Stochastic gradient descent, SGD)」という方法が使われます。
「確率的勾配法」は、局所解問題の解法の一つでもあります。

・勾配法をより洗練させたものとして、2015年にキングマン氏等が発表した
「Adam (Adaptive moment estimation)」というアルゴリズムもあります。
これ以外にも、モデルの最適化（誤差関数の値を減少させるためのパラメータの最適化）のために
勾配法を改良した様々な方法が発表されています。

※　誤差逆伝搬法は、
本セミナー第８回目「機械学習（２回目：機械学習の概要と教師あり学習（分類））」
で、改めて取り上げます。

【出典・参考】

⇒「ﾃﾞｨｰﾌﾟﾗｰﾆﾝｸﾞがわかる数学入門」涌井良幸・貞美著 技術評論社 2017.04
⇒「Pythonで動かして学ぶ！あたらしい機械学習の教科書」（2018年01月 翔泳社 伊藤真著）
⇒ 勾配法 https://qiita.com/tokkuman/items/1944c00415d129ca0ee9
⇒ 勾配降下法　http://shironeko.hateblo.jp/entry/2016/10/29/173634
⇒ 最急降下法　https://ja.wikipedia.org/wiki/最急降下法
⇒ 誤差逆伝播法 https://ja.wikipedia.org/wiki/バックプロパゲーション
⇒ 確率的勾配降下法 https://ja.wikipedia.org/wiki/確率的勾配降下法
⇒ 最適化 http://www.orsj.or.jp/archive2/or60-4/or60_4_191.pdf

(6) 確認問題

以下の空欄に最もあてはまる回答を選択肢から選び、その記号を回答欄に記入してください。

(1) 平均変化率・微分可能・導関数・微分係数

・区間 J（例えば -100≦x≦100 等）で定義された関数 y = f(x) について、
変数 x（x ∈ J） が x から x+⊿x まで変化する時、y が y から y+⊿y まで変化したとします。この時、x の増分⊿x に対する y の増分は ⊿y で、
⊿y/⊿x は、y = f(x) の(1.1)＿＿＿＿＿＿といい、次式で与えられます：

$$\Delta y/\Delta x = (f(x+\Delta x) - f(x)) /\Delta x$$

・(1.1)＿＿＿＿＿＿について、点 x における増分⊿xを限りなく 0 に近づけた場合に、
(1.1)＿＿＿＿＿＿の有限な極限値が、区間 J 内の各点で存在する時、
関数 y = f(x) は区間 J で(1.2)＿＿＿＿＿＿である、と言います。

その極限値を、関数 y = f(x) の(1.3)＿＿＿＿＿＿と呼び、
f'(x) または df(x)/dx と表現します。
また、関数 y = f(x) の(1.3)＿＿＿＿＿＿を求めることを、
関数 f(x) を(1.4)＿＿＿＿＿＿と言います。

$$df(x)/dx = f'(x) = \lim_{\Delta x \to 0} (f(x+\Delta x) - f(x)) /\Delta x$$

・変数 x = a における関数 y = f(x) の(1.3)＿＿＿＿＿＿の値 f'(a) を
関数 y = f(x) の x = a における(1.5)＿＿＿＿＿＿と呼び、
f'(a) または df(a)/dx と表現します。

$$df(a)/dx = f'(a)$$

・変数 x = a における微分係数 f'(a) は、y = f(x) のグラフ上では、
点(a,f(a))における(1.6)＿＿＿＿＿＿の傾きを表しています。

（選択肢）

(a) 「導関数（どうかんすう、derived function 又は derivative)」
(b) 「微分する（differentiate)」
(c) 「微分可能（ﾋﾞﾌﾞﾝｶﾉｳ、differentiable)」
(d) 「微分係数（ﾋﾞﾌﾞﾝｹｲｽｳ、differential coefficient)」
(e) 「平均変化率（ﾍｲｷﾝｼﾞﾍﾝｶﾘﾂ、average rate of change)」
(f) 接線

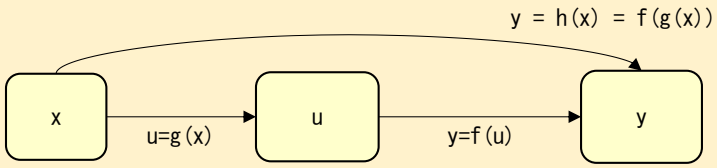
（回答）

(1.1)	
(1.2)	
(1.3)	
(1.4)	
(1.5)	
(1.6)	

(2) 合成関数・逆関数の微分

(2-1) 合成関数の微分

y が変数 u の関数 y = f(u) で、
u が変数 x の関数 u = g(x) の時、
y は変数 x の関数 y = h(x) となります。



関数h は、関数g と関数f との「合成関数（ごうせいかんすう、composite function)」
と言い、次のように表現します。

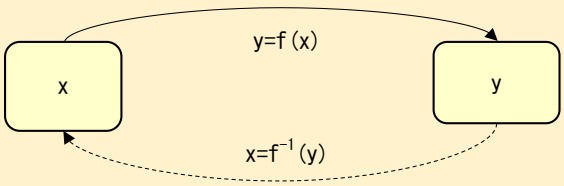
$$y = h(x) = f(g(x))$$

関数g と関数f とが微分可能である時、
関数g と関数f との合成関数h の導関数は、それぞれの導関数の積で与えられます。
これを(2.1)と言います。

$$dy/dx = \text{(2.2)}$$

(2-2) 逆関数の微分

x の関数 y = f(x) が1対1の写像である時、
関数 y = f(x) の「逆関数（ぎゃくかんすう、inverse function)」が存在し、
$x = f^{-1}(y)$ で表現します。
逆関数について、$f^{-1}(f(x)) = x$ が成立します。



関数 y = f(x) が微分可能で、$df(x)/dx \neq 0$ ならば、
その逆関数 $x = f^{-1}(y)$ もまた微分可能で、
その導関数は次式で与えられます。

$$dx/dy = 1 \,/\, dy/dx$$

なお、関数としては一般に従属変数y を独立変数x の関数として表現するので、
逆関数もそのように書きかえて(x→y, y→x に書きかえて)扱い、
逆関数の微分は以下の様になります。

$$dy/dx = \text{(2.3)}$$

（選択肢）

(a) $dy/dx = (dy/du)/(dx/du)$
(b) $dy/dx = dy/du \cdot du/dx$
(c) $dy/dx = 1 \,/\, dx/dy$
(d) 「結合律（ｹﾂｺﾞｳﾘﾂ、associative law)」
(e) 「交換律（ｺｳｶﾝﾘﾂ、commutative law)」
(f) 「連鎖律（ﾚﾝｻﾘﾂ、chain rule)」

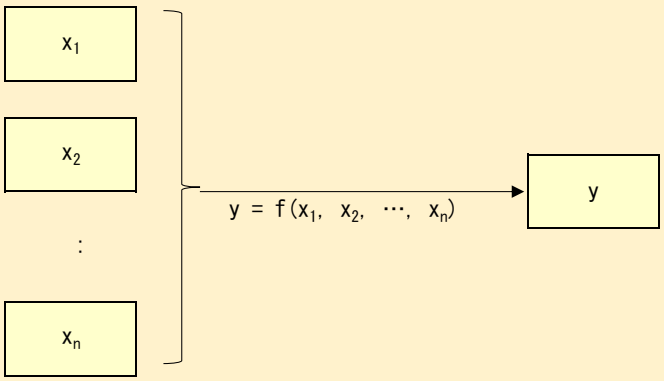
（回答）

(2.1)	
(2.2)	
(2.3)	

(3) 多変数関数と偏微分

(3-1) 多変数関数

・変数 y が複数の変数(x_1, x_2, …, x_n)の関数f となっているとき、
関数f を「多変数関数 (ﾀﾍﾝｽｳｶﾝｽｳ、multivariable function)」と言います。



(3-2) 偏微分

・n個の変数(x_1, x_2, …, x_n)の多変数関数 $y = f(x_1, x_2, \cdots, x_n)$ において、
一つの変数 x_k (k=1,2,…,n) 以外を固定すれば、

$$y = f(x_1, x_2, \cdots, x_n)$$

は、一つの変数 x_k だけの関数となります。
この変数 x_k の関数の導関数

$$\lim_{h \to 0} \left(f(x_1, x_2, \cdots, x_k+h, \cdots, x_n) - f(x_1, x_2, \cdots, x_k, \cdots, x_n) \right) \Big/ h$$

が存在する時、
関数 $y = f(x_1, x_2, \cdots, x_n)$ は、x_k で (3.1) ＿＿＿＿＿＿＿＿
であるといい、その極限値としてできた (x_1, x_2, …, x_n) の新しい関数を
関数 $y = f(x_1, x_2, \cdots, x_n)$ のx_k に関する (3.2) ＿＿＿＿＿＿＿＿
と呼び、 以下の様に表現します。

$$f_{xk}(x_1, x_2, \cdots, x_n)$$

または $\partial f(x_1, x_2, \cdots, x_n) / \partial x_k$ （∂は「ﾃﾞﾙ」と呼びます）

・また、多変数関数 $y = f(x_1, x_2, \cdots, x_n)$ の x_k に関する偏導関数を求めることを、
関数 $y = f(x_1, x_2, \cdots, x_n)$ を x_k で(3.3) ＿＿＿＿＿＿＿＿ する、と言います。

（選択肢）

(a) 「全微分可能 (ｾﾞﾝﾋﾞﾌﾞﾝｶﾉｳ、total differentiable)」
(b) 「全微分 (ｾﾞﾝﾋﾞﾌﾞﾝ、total derivative)」
(c) 「偏導関数 (ﾍﾝﾄﾞｳｶﾝｽｳ、partial derived function)」
(d) 「偏微分 (ﾍﾝﾋﾞﾌﾞﾝ、partial differentiate)」
(e) 「偏微分可能 (ﾍﾝﾋﾞﾌﾞﾝｶﾉｳ、partial differentiable)」

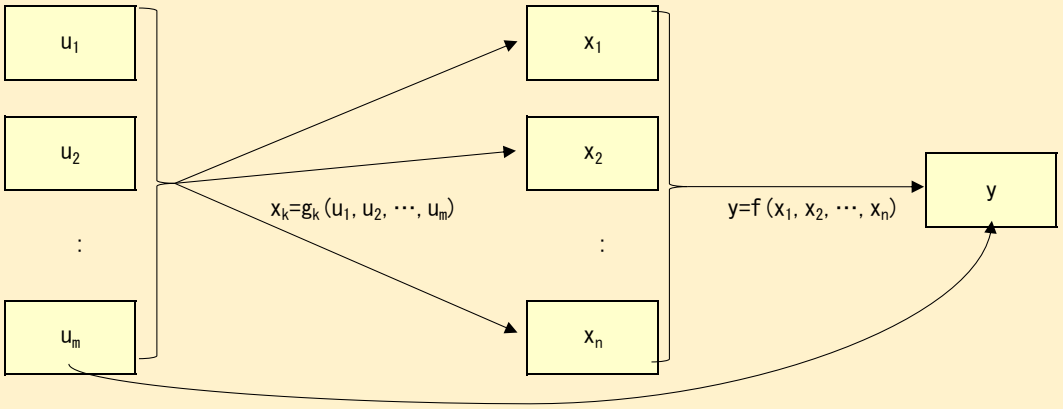
（回答）

(3.1)	
(3.2)	
(3.3)	

(4) 合成関数の偏微分

(4-1) 合成関数の偏微分

n個の変数$(x_1, x_2, \cdots, x_n)$の多変数関数 $y=f(x_1, x_2, \cdots, x_n)$ において、
変数 x_k $(k=1, 2, \cdots, n)$ が、m個の変数$(u_1, u_2, \cdots, u_m)$の多変数関数 $x_k=g_k(u_1, u_2, \cdots, u_m)$
となっているとします。



この時、n個の変数$(x_1, x_2, \cdots, x_n)$の多変数関数 $y=f(x_1, x_2, \cdots, x_n)$ は、
変数$(u_1, u_2, \cdots, u_m)$ の関数として、次式により与えられます：

$$y = f(g_1(u_1, u_2, \cdots, u_m),\ g_2(u_1, u_2, \cdots, u_m),\ \cdots,\ g_n(u_1, u_2, \cdots, u_m)) \qquad \cdots(式4.3\text{-}4)$$

この偏導関数に関して、
関数 $y=f(x_1, x_2, \cdots, x_n)$ の x_k $(k=1, 2, \cdots, n)$ に関する偏導関数 $\partial f/\partial x_k$ が連続で、
関数 $x_k=g_k(u_1, u_2, \cdots, u_m)$ が u_j $(j=1, 2, \cdots, m)$ について偏微分可能ならば、
その偏導関数は、次式により与えられます：

$$\begin{aligned} \partial y/\partial u_j &= \partial f/\partial x_1 \cdot \partial x_1/\partial u_j \\ &\quad + \partial f/\partial x_2 \cdot \partial x_2/\partial u_j \\ &\quad + \cdots \\ &\quad + \partial f/\partial x_n \cdot \partial x_n/\partial u_j \end{aligned} \qquad (j=1, 2, \cdots, m)$$

または、

$$\partial y/\partial u_j = \sum_{k=1}^{n} \partial f/\partial x_k \cdot \partial x_k/\partial u_j \qquad (j=1, 2, \cdots, m)$$

または、

$$\partial y/\partial u_j = \sum_{k=1}^{n} \partial f/\partial g_k \cdot \partial g_k/\partial u_j \qquad (j=1, 2, \cdots, m)$$

これが合成関数の偏微分についての (4.1) ＿＿＿＿ です。

（選択肢）

(a) 「結合律 (ｹﾂｺﾞｳﾘﾂ、associative law)」
(b) 「交換律 (ｺｳｶﾝﾘﾂ、commutative law)」
(c) 「連鎖律 (ﾚﾝｻﾘﾂ、chain rule)」

（回答）

(4.1)	

(5) 多変数関数の近似公式と勾配

・n個の変数$(x_1,\ x_2,\ \cdots,\ x_n)$ の多変数関数 $y = f(x_1,\ x_2,\ \cdots,\ x_n)$ において、
変数$(x_1,\ x_2,\ \cdots,\ x_n)$ から少し変化させた$(x_1+\Delta x_1,\ x_2+\Delta x_2,\ \cdots,\ x_n+\Delta x_n)$での
多変数関数 $y = f(x_1+\Delta x_1,\ x_2+\Delta x_2,\ \cdots,\ x_n+\Delta x_n)$の近似値は、次式で与えられます。

$$
\begin{aligned}
&f(x_1+\Delta x_1,\ x_2+\Delta x_2,\ \cdots,\ x_n+\Delta x_n) && \cdots(式①)\\
&\fallingdotseq f(x_1,\ x_2,\ \cdots,\ x_n)\\
&+\ \partial f(x_1, x_2, \cdots, x_n)/\partial x_1 \cdot \Delta x_1 && (\Delta x_1は小さな数)\\
&+\ \partial f(x_1, x_2, \cdots, x_n)/\partial x_2 \cdot \Delta x_2 && (\Delta x_2は小さな数)\\
&+\ \cdots\\
&+\ \partial f(x_1, x_2, \cdots, x_n)/\partial x_n \cdot \Delta x_n && (\Delta x_nは小さな数)
\end{aligned}
$$

・これは点$(x_1,\ x_2,\ \cdots,\ x_n)$ での関数値 $f(x_1,\ x_2,\ \cdots,\ x_n)$ に、
各変数x_k(k=1～n)毎の（変化率×変化量）の総和を加えたものになっています。

・実は、これは (5.1)＿＿＿＿＿ において、
Δx_k(k=1～n)の一次の項までを取り上げたものに相当します。

・多変数関数の近似公式は、ベクトルとその内積で表現することが出来ます。
はじめに、n個の変数$(x_1,\ x_2,\ \cdots,\ x_n)$ の多変数関数 $y = f(x_1,\ x_2,\ \cdots,\ x_n)$ の、
各成分ごとの偏微分を並べた (5.2)＿＿＿＿＿というベクトルを
導入します。

(5.2)

$$\nabla f = \left(\partial f/\partial x_1,\ \partial f/\partial x_2,\ \cdots,\ \partial f/\partial x_n\right)$$

次に、点$(x_1,\ x_2,\ \cdots,\ x_n)$ から変化させた点$(x_1+\Delta x_1,\ x_2+\Delta x_2,\ \cdots,\ x_n+\Delta x_n)$への
移動量$(\Delta x_1,\ \Delta x_2,\ \cdots,\ \Delta x_n)$を移動量ベクトル$\boldsymbol{\Delta x}$で表現します。

移動量ベクトル

$$\boldsymbol{\Delta x} = (\Delta x_1,\ \Delta x_2,\ \cdots,\ \Delta x_n)$$

(式①)で表現された多変数関数 $y = f(x1,\ x2,\ \cdots,\ xn)$の変化量$\Delta y$ は、
これらのベクトルの内積になります。

関数 $f(x_1,\ x_2,\ \cdots,\ x_n)$ の変化量Δyのベクトル表現

$$
\begin{aligned}
\Delta y\ &= f(x_1+\Delta x_1,\ x_2+\Delta x_2,\ \cdots,\ x_n+\Delta x_n) - f(x_1,\ x_2,\ \cdots,\ x_n) && \cdots(式①')\\
&\fallingdotseq \partial f(x_1, x_2, \cdots, x_n)/\partial x_1 \cdot \Delta x_1 && (\Delta x_1は小さな数)\\
&+\ \partial f(x_1, x_2, \cdots, x_n)/\partial x_2 \cdot \Delta x_2 && (\Delta x_2は小さな数)\\
&+\ \cdots\\
&+\ \partial f(x_1, x_2, \cdots, x_n)/\partial x_n \cdot \Delta x_n && (\Delta x_nは小さな数)\\
\therefore\ \Delta y\ &\fallingdotseq \nabla f \cdot \boldsymbol{\Delta x} && \cdots(式②)
\end{aligned}
$$

（選択肢）

(a) 「▽（ﾅﾌﾞﾗ、nabla, del)」
(b) 「テイラー展開 (Taylor's expansion)」
(c) 「フーリエ展開 (Fourier's expansion)」
(d) 「勾配（ｺｳﾊﾞｲ、ｸﾞﾗﾃﾞｨｴﾝﾄ、gradient)」

（回答）

(5.1)	
(5.2)	

(6) 勾配降下法

・n個の変数(x_1, x_2, …, x_n) の多変数関数 y = f(x_1, x_2, …, x_n) について、
点(x_1, x_2, …, x_n) での関数 f の値と、
点(x_1, x_2, …, x_n) から移動量ベクトル(Δx_1, Δx_2, …, Δx_n)だけ変化させた
点($x_1+\Delta x_1$, $x_2+\Delta x_2$, …, $x_n+\Delta x_n$) での関数 f の値との変化量Δy は、
勾配ベクトル∇f と移動量ベクトルΔx の内積になります。

関数 f(x_1, x_2, …, x_n) の変化量Δy

$$\Delta y = \nabla f \cdot \Delta \mathbf{x}$$

∇f は勾配ベクトル ($\partial f/\partial x_1$, $\partial f/\partial x_2$, ・・・, $\partial f/\partial x_n$)

$\Delta \mathbf{x}$ は移動量ベクトル (Δx_1, Δx_2, …, Δx_n)

・この関係式から、
関数 f(x_1, x_2, …, x_n)の変化量Δy を最小にするのは（つまり、関数f の値を最も減少させるのは）、
勾配ベクトル∇f と移動量ベクトルΔx の内積を最小にする、ということと同じになります。

内積が最小になるのは、勾配ベクトル∇f と移動量ベクトルΔx とが反対向きの場合です。
従って、点(x_1, x_2, …, x_n) から移動量ベクトル$\Delta \mathbf{x}$=(Δx_1, Δx_2, …, Δx_n)だけ変化させた
点($x_1+\Delta x_1$, $x_2+\Delta x_2$, …, $x_n+\Delta x_n$)へ移動する時、
関数 f(x_1, x_2, …, x_n) の値が最も減少するのは、次の関係が満たされる時になります。

関数 f(x_1, x_2, …, x_n) の値を最も小さくする移動量ベクトル$\Delta \mathbf{x}$

$$\Delta \mathbf{x} = -\eta \nabla f$$

η は「学習係数、または、学習率 (ｶﾞｸｼｭｳｹｲｽｳ, ｶﾞｸｼｭｳﾘﾂ, learning rate)」
と呼ばれる正の小さな定数

この移動量ベクトル$\Delta \mathbf{x}$を採用することにより、
関数 f(x_1, x_2, …, x_n) を減少させることが出来ます。
すなわち、以下の様な手順を踏んで、関数 f(x_1, x_2, …, x_n)を減少させることが出来ます。
これが (6.1) あるいは、
「最急降下法 (ｻｲｷｭｳｺｳｶﾎｳ, steepest descent method)」と呼ばれる方法です。

勾配降下法の基本的アルゴリズム

(1) iを繰り返しの通し番号として、i=1 から開始する。
(2) 学習係数η (>0) を決定する(※1)。
(3) 初期位置$\mathbf{x}_i$=(x_{i1}, x_{i2}, …, x_{in}) を決定する。

(4) 以下の様な繰り返し終了判定を行う。
(4.1) 点$\mathbf{x}_i$ での関数 f($\mathbf{x}_i$) の値が十分小さければ、本処理を終了する。
この時は 点$\mathbf{x}_i$ が関数 f の最小値を与える点と判断する。
(4.2) 上記(4.1)以外で、繰り返し回数が制限を超過した場合、本処理を終了する。
この時は 最小値を与える点は見つかっておらず、モデル等の見直しを行う必要あり。
(4.3) 上記(4.1)、(4.2)以外の場合、処理(5)へ。

(5) 点$\mathbf{x}_i$ からの移動量ベクトル$\Delta \mathbf{x}$ を(式5.3-1)により計算して点$\mathbf{x}_{i+1}$ へ移動する。
$\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta \mathbf{x}$
ここで、$\Delta \mathbf{x} = -\eta (\partial f(\mathbf{x}_i)/\partial x_1, \partial f(\mathbf{x}_i)/\partial x_2, ・・・, \partial f(\mathbf{x}_i)/\partial x_n)$

(6) i ← i+1 として手順(4)へ

(※1) 学習係数ηを、手順(4)～(6)の過程で動的に決定する方法もあります。

・以下の図は多変数関数 y と、その勾配降下法による最小値探索の例です。

$$y(x_1, x_2) = x_1^2 + x_2^2 + 3x_2 + 4$$

この関数が構成する面の勾配ベクトル∇y は、

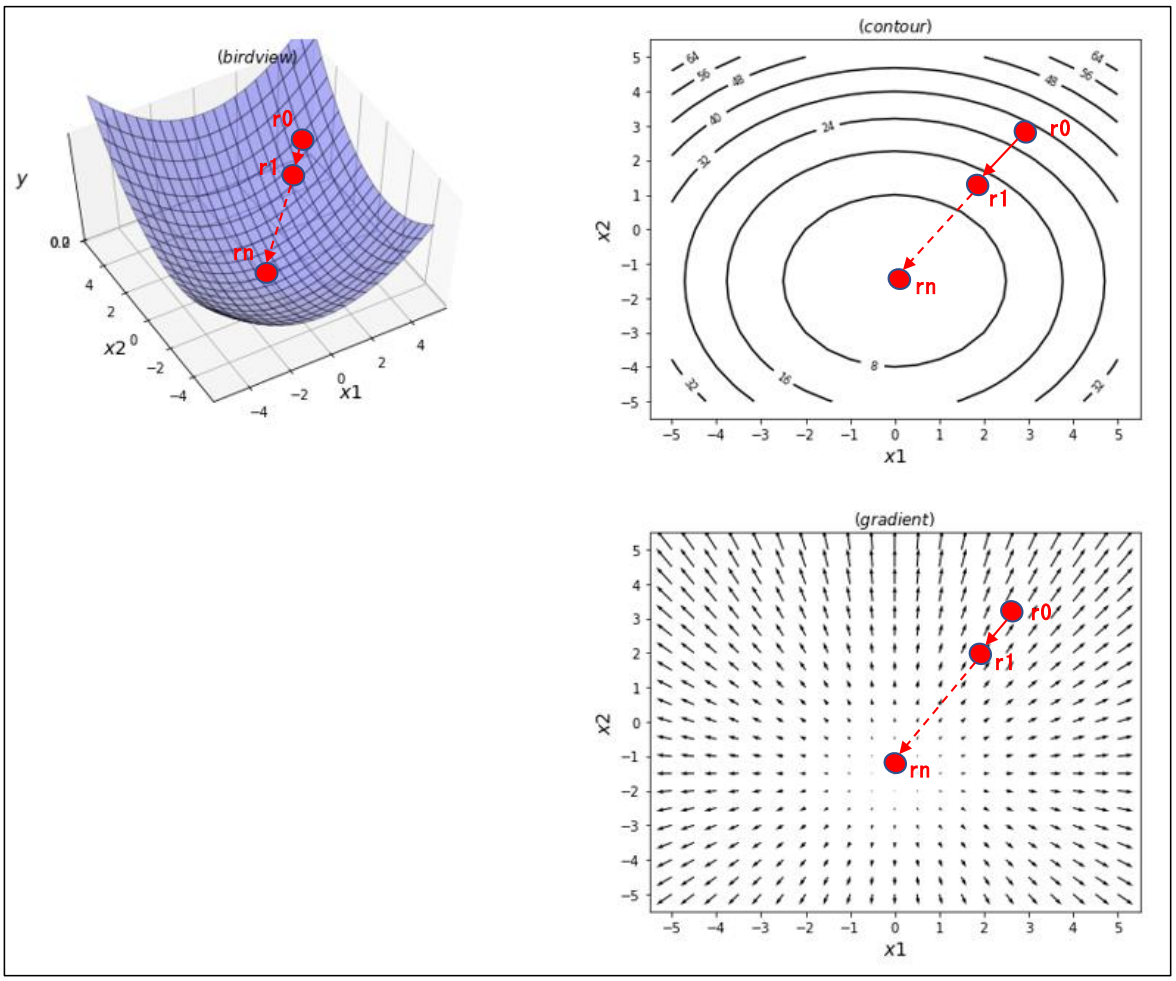
$$\nabla y = (\partial y/\partial x_1,\ \partial y/\partial x_2) = (2x_1,\ 2x_2+3)$$

で与えられ、それを示したものは右下図（gradient）です。

下図では、丸点とそれをつなぐ矢印は、初期位置 $\mathbf{r}_0$ から始まって、

$$\mathbf{r}_{i+1} = \mathbf{r}_i - \eta(\partial y/\partial x_1,\ \partial y/\partial x_2) \qquad \eta\ (>0)：学習係数$$

という移動を繰り返して、関数値が最小の点 $\mathbf{r}_n$ に落ち着く様子を示しています。



・尚、(6.1)＿＿＿＿＿では、極小値が最小値とは限らないのに、
最小値でない極小値にはまってしまうと抜け出られない場合がある、という欠点があります。
これが (6.2)＿＿＿＿＿ であり、
(6.1)＿＿＿＿＿ を改良した様々な方法が提案されています。

（選択肢）

(a) 「勾配降下法（ｺｳﾊﾞｲｺｳｶﾎｳ, gradient descent method）」
(b) 「擬似焼きなまし法（ｷﾞｼﾞﾔｷﾅﾏｼﾎｳ、Simulated annealing）」
(c) 「局所解問題（ｷｮｸｼｮｶｲﾓﾝﾀﾞｲ、local minumum problem）」
(d) 「勾配消失問題（ｺｳﾊﾞｲｼｮｳｼﾂﾓﾝﾀﾞｲ、vanishing gradient problem）」

（回答）

(6.1)	
(6.2)	

(7) 基本的な関数の微分

(7-1) べき関数の導関数

・べき関数の導関数

$$d\ x^{\alpha}/dx \quad = \quad (7.1) \qquad (\alpha \text{は実数})$$

(7-2) 指数関数の導関数

・aを底とする指数関数の導関数

$$d\ a^{x}/dx \quad = \quad (7.2) \qquad (a>0, a\neq 1)$$

・eを底とする指数関数の導関数

$$d\ e^{x}/dx \quad = \quad (7.3) \qquad (e\text{はネイピア数})$$

(7-3) 対数関数の導関数

・aを底とする対数の導関数

$$d\ \log_a(x)/dx \quad = \quad (7.4) \qquad (a>0, a\neq 1)$$

・自然対数の導関数

$$d\ \log(x)/dx \quad = \quad (7.5)$$

(7-4) 三角関数の導関数

・正弦 (sine)関数の導関数

$$d\ \sin(x)/dx \quad = \quad (7.6)$$

・余弦 (cosine)関数の導関数

$$d\ \cos(x)/dx \quad = \quad (7.7)$$

・正接 (tangent)関数の導関数

$$d\ \tan(x)/dx \quad = \quad (7.8)$$

・余割 (cosecant)関数の導関数

$$d\ \mathrm{cosec}(x)/dx \quad = \quad -\mathrm{cosec}(x)\cot(x)$$

・正割 (secant)関数の導関数

$$d\ \sec(x)/dx \quad = \quad \sec(x)\tan(x)$$

・余接 (cotangent)関数の導関数

$$d\ \cot(x)/dx \quad = \quad -\mathrm{cosec}^2(x)$$

(7-5) 逆三角関数の導関数

・逆正弦 (arc sine)関数の導関数

$$d\ \sin^{-1}(z)/dz \quad = \quad 1/\sqrt{1-z^2} \qquad (-1<x<1)$$

・逆余弦 (arc cosine)関数の導関数

$$d\ \cos^{-1}(z)/dz \quad = \quad -1/\sqrt{1-z^2} \qquad (-1<x<1)$$

・逆正接 (arc tangent)関数の導関数

$$d\ \tan^{-1}(z)/dz \quad = \quad 1/(1+z^2)$$

・逆余接 (arc cotangent)関数の導関数

$$d\ \cot^{-1}(z)/dz \quad = \quad -1/(1+z^2)$$

（選択肢）

(a) $1/x$
(b) $1/x\log(a)$
(c) $a^x \log(a)$
(d) $\cos(x)$
(e) e^x
(f) $\sec^2(x)$
(g) $-\sin(x)$
(h) $\alpha x^{\alpha-1}$

（回答）

(7.1)	
(7.2)	
(7.3)	
(7.4)	
(7.5)	
(7.6)	
(7.7)	
(7.8)	

(7) 確認問題回答用紙

提出者　：
提出日　：　　年　　月　　日

回答

No.	回答
(1.1)	
(1.2)	
(1.3)	
(1.4)	
(1.5)	
(1.6)	
(2.1)	
(2.2)	
(2.3)	
(3.1)	
(3.2)	
(3.3)	
(4.1)	
(5.1)	
(5.2)	
(6.1)	
(6.2)	

No.	回答
(7.1)	
(7.2)	
(7.3)	
(7.4)	
(7.5)	
(7.6)	
(7.7)	
(7.8)	

（全 25 問）

（※黄色の枠　のみ記入をお願いします）

※ ご意見・ご要望など有りましたら、下欄に記してください。