

# 数学で理解するディープラーニング

教科書：

巣籠悠輔著，詳解ディープラーニング TensorFlow・Kerasによる時系列データ処理，マイナビ出版，2017

馬場・小島・小澤著, ニューラルネットワークの基礎と応用, 共立出版，1994

# 第1章　数学の準備

## 1-0 常微分

1. 関数

Ex1 のとき次の値を求めよ。

1. 平均変化率

Ex2 次の関数のの平均変化率を求めよ。

1. 導関数

Ex3 次の関数を導関数の定義に基いて微分せよ。

1. 積の微分

Ex4 次の関数を微分せよ。

1. 合成関数の微分

Ex5 の時、次の関数を微分せよ。

1. 微分公式

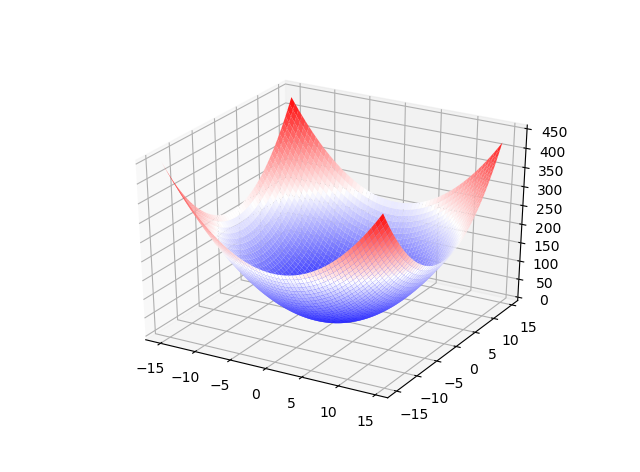
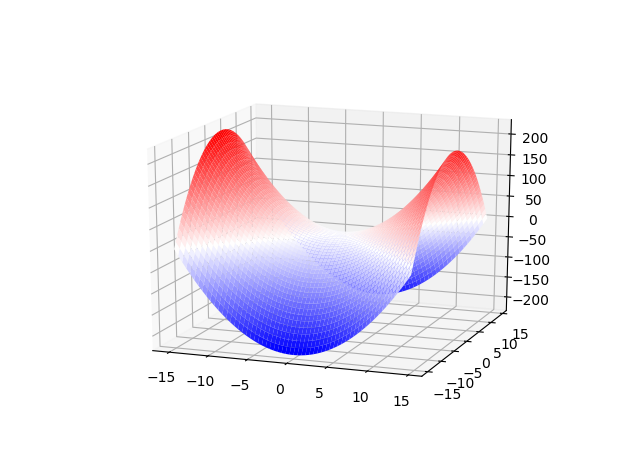
Ex6 次の関数を微分せよ。

Ex7 次の関数を微分せよ。

* + 1. シグモイド関数
    2. ソフトマックス関数

## 1-1偏微分

(1) 多変数関数



例

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import axes3d

fig = plt.figure()

ax = fig.add\_subplot(111, projection=’3d’)

x = y = np.arange(-15, 15, 0.5)

X, Y = np.meshgrid(x, y)

Z = X\*\*2 – Y\*\*2

ax.plot\_surface(X, Y, Z, rstride=1, cstride=1, cmap=’bwr’)

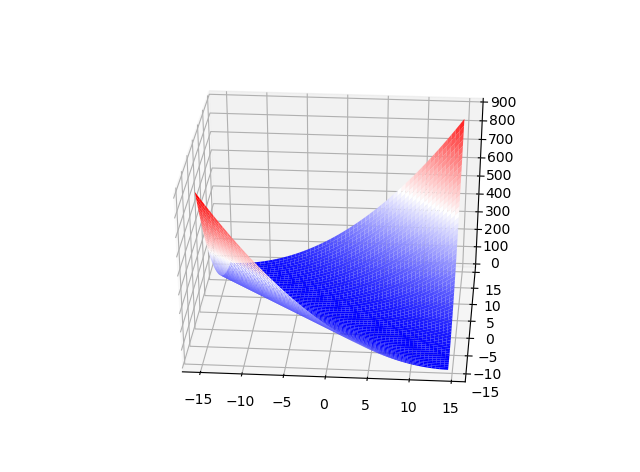
plt.show()

(2) 偏微分

複数の変数のうちどれか1つの変数のみに関して微分すること。他の変数は定数とみなす。

例　 をについて偏微分すると　　　

Ex7 のとき、



(3) 連鎖律

Ex8 のとき、

## 1-2 線形代数

1. ベクトルの内積

Ex9

のとき

1. 行列

Ex10  
のとき、

# 第2章　Pythonの準備

Ex1

を計算する関数を作れ

Ex2 フィボナッチ数列を求め、リストにせよ

Ex3  
 のとき、

Ex1 解答例----------------------------------------------------------------------------------------------------------

import numpy as np

A = np.array([[1, 2], [3, 4]])

B = np.array([[5, 6, 7], [8, 9, 10]])

AB = np.dot(A, B)

BtA = np.dot(B.T, A)

# BtA = np.dot(B, A)　だとダメ

print('AB = ', AB)

print('BtA = ', BtA)

----------------------------------------------------------------------------------------------------------------

# 第3章　ニューラルネットワーク

## 3-1 単純パーセプトロン

### (1) 線形重回帰モデル

ここで、

とおくと

と書ける。

いま、すなわち、

とすると、

なので、勾配ベクトルの成分は、

となる。この勾配ベクトルの逆向きに解を探索していく。

すなわち、学習率（learning rate）をとして、

いま、とすると、

となり、テキストの式(3.17)を得る。

学習の次*ステップの*重み**’**と*バイアス*は、

Ex1　前ページ～このページの各式の右辺を埋めよ。

Ex1 解答----------------------------------------------------------------------------------------------------------------------

いま、とすると、

となり、テキストの式(3.17)を得る。

学習の次*ステップの*重み**’**と*バイアス*は、

-----------------------------------------------------------------------------------------------------------------------

Ex2　テキストpp.083-085を参考に図3.11を出力するプログラムを作れ。

Ex2　解答例　--------------------------------------------------------------------------------------------------------------

import numpy as np

import matplotlib.pyplot as plt

rng = np.random.RandomState(123)

#rng = np.random

d = 2

N = 10

mean = 5

x1 = rng.randn(N, d) + np.array([0, 0])

x2 = rng.randn(N, d) +np.array([mean, mean])

x = np.concatenate((x1, x2), axis=0)

X = x[:,0]

Y = x[:,1]

plt.scatter(X, Y)

w = np.zeros(d)

b = 0

def y(x):

return step(np.dot(w, x) + b)

def step(x):

return 1 \* (x > 0)

def t(i):

if i < N:

return 0

else:

return 1

while True:

classified = True

for i in range(N\*2):

delta\_w = (t(i) - y(x[i])) \* x[i]

delta\_b = t(i) - y(x[i])

w += delta\_w

b += delta\_b

classified \*= all(delta\_w == 0) \* (delta\_b == 0)

if classified:

break

print(w[0])

print(w[1])

print(b)

X = range(0, 10, 2)

Y = - w[0] \* X / w[1] - b / w[1]

plt.plot(X, Y, 'b')

plt.show()

-----------------------------------------------------------------------------------------------------------------------------------

Ex3　2つのグループのドットを色分けして表示するように上のプログラムを変更せよ。

Ex4 ANDゲートのモデルを作成せよ。

Ex5　ORゲートのモデルを作成せよ。

### (2) ロジスティック回帰

ここで、

とし、すなわち、

とすると、

また、

なので、勾配ベクトルの成分は、

となるので、この勾配ベクトルの逆向きに解を探索していく。

すなわち、学習率（learning rate）をとして、

よって、学習の次*ステップの*重み**’**と*バイアス*は、

Ex6　前ページの各式の右辺を埋めよ。

Ex6 解答-----------------------------------------------------------------------------------------------------------------------

となるので、この勾配ベクトルの逆向きに解を探索していく。

すなわち、学習率（learning rate）をとして、

よって、学習の次*ステップの*重み**’**と*バイアス*は、

-----------------------------------------------------------------------------------------------------------------------------------

ただしこの方法は計算負荷が大きく実用的ではない。

そこで、計算負荷軽減のため、確率的勾配降下法やミニバッチ勾配降下法を用いることが多い。

#### 確率的勾配降下法(stochastic gradient descent, SGD)

からランダムに1つ選んだ自然数として、

とすることで、テキストの式(3.38), (3.39)を得る。

すなわち、学習の次*ステップの*重み**’**と*バイアス*は、

#### ミニバッチ勾配降下法(mini-batch gradient descent)

勾配降下法とSGDの中間的な方法が、ミニバッチ勾配降下法(mini-batch gradient descent)である。

Ex5　このページの各式の右辺を埋めよ。