追加レポート：1-3.出力層

実装演習のコードで、「試してみよう」の部分を実装し出力結果を確認する。

In [35]:
```
import numpy as np
from common import functions
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')
```

In [36]:
```
def print_vec(text, vec):
    print(f'{text} :\n{vec}')
```

順伝播（3層・複数ユニット）

In [37]:
```
def init_network():
    """重みとバイアスを設定し、ネットワークを作成する。"""
    print("##### ネットワークの初期化 #####")
    network = {}

    #試してみよう
    #_各パラメータのshapeを表示
    #_ネットワークの初期値ランダム生成

    network['W1'] = np.random.rand(2, 3)
    network['W2'] = np.random.rand(3, 2)
    network['W3'] = np.random.rand(2, 2)

    network['b1'] = np.random.rand(3)
    network['b2'] = np.random.rand(2)
    network['b3'] = np.random.randint(1, 5, size=2)

    print_vec('重み1', network['W1'] )
    print_vec('重み1のサイズ', network['W1'].shape)
    print_vec('重み2', network['W2'] )
    print_vec('重み2のサイズ', network['W2'].shape)
    print_vec('重み3', network['W3'] )
    print_vec('重み3のサイズ', network['W3'].shape)
    print_vec('バイアス1', network['b1'] )
    print_vec('バイアス1のサイズ', network['b1'].shape)
    print_vec('バイアス2', network['b2'] )
    print_vec('バイアス2のサイズ', network['b2'].shape)
    print_vec('バイアス3', network['b3'] )
    print_vec('バイアス3のサイズ', network['b3'].shape)

    return network
```

In [38]:
```
def forward(network, x):
    """プロセスを作成"""
    print("##### 順伝播開始 #####")

    W1, W2, W3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']

    # 1層の総入力
    u1 = np.dot(x, W1) + b1

    # 1層の総出力(ReLU)
    z1 = functions.relu(u1)

    # 2層の総入力
    u2 = np.dot(z1, W2) + b2

    # 2層の総出力(ReLU)
    z2 = functions.relu(u2)

    # 出力層の総入力
    u3 = np.dot(z2, W3) + b3

    # 出力層の総出力
    y = u3

    print_vec("総入力1", u1)
    print_vec("中間層出力1 (ReLU)", z1)
    print_vec("総入力2", u2)
    print_vec("中間層出力2 (ReLU)", z2)
    print_vec("出力層の総入力", u3)
    print(f"出力合計: {np.sum(y)}")

    return y, z1, z2
```

In [39]:
```
# 入力値
x = np.array([1., 2.])
print_vec("入力値", x)
```

```
入力値：
[1. 2.]
```

In [40]:
```
# ネットワークの初期化
network =  init_network()
```

```
##### ネットワークの初期化 #####
重み1：
[[0.45033305 0.96499452 0.68881195]
 [0.98684881 0.06896795 0.49631945]]
重み1のサイズ：
(2, 3)
重み2：
[[0.8112913  0.21245298]
 [0.70642581 0.37683864]
 [0.68654945 0.88066562]]
重み2のサイズ：
(3, 2)
重み3：
[[0.19860226 0.69034063]
 [0.82125133 0.18286657]]
重み3のサイズ：
(2, 2)
バイアス1：
[0.17479751 0.79781699 0.17574085]
バイアス1のサイズ：
(3,)
バイアス2：
[0.26267015 0.68604678]
バイアス2のサイズ：
(2,)
バイアス3：
[1 4]
バイアス3のサイズ：
(2,)
```

In [41]:
```
# 順伝搬
y, z1, z2 = forward(network, x)
```

```
##### 順伝播開始 #####
総入力1：
[2.59882818 1.90074742 1.85719171]
中間層出力1 (ReLU)：
[2.59882818 1.90074742 1.85719171]
総入力2：
[4.98886782 3.59001554]
中間層出力2 (ReLU)：
[4.98886782 3.59001554]
出力層の総入力：
[4.93910545 8.10051199]
出力合計: 13.039617442510599
```

多クラス分類（3-5-4ネットワーク）

！試してみよう_ノードの構成を 3-5-4 に変更してみよう

In [42]:
```
def init_network():
    print('###ネットワーク初期化###')

    network = {}

    network['W1'] = np.random.rand(3, 5)
    network['W2'] = np.random.rand(5, 4)
    network['b1'] = np.random.rand(1, 5)
    network['b2'] = np.random.rand(1, 4)

    print_vec('重み1', network['W1'] )
    print('W1.shape: {}'.format(network['W1'].shape))
    print_vec('重み2', network['W2'] )
    print('W2.shape: {}'.format(network['W2'].shape))
    print_vec('バイアス1', network['b1'] )
    print('b1.shape: {}'.format(network['b1'].shape))
    print_vec('バイアス2', network['b2'] )
    print('b2.shape: {}'.format(network['b2'].shape))

    return network
```

In [43]:
```
def forward(network, x):
    print('###順伝搬を開始###')
    W1, W2 = network['W1'], network['W2']
    b1, b2 = network['b1'], network['b2']

    # 1層の総入力
    u1 = np.dot(x, W1) + b1
    # 1層の総出力
    z1 = functions.relu(u1)

    # 2層の総入力
    u2 = np.dot(z1, W2) + b2

    # 出力層の総出力（softmax関数）
    y = functions.softmax(u2)

    print_vec('総入力1', u1)
    print_vec('中間層出力1', z1)
    print_vec('総入力2', u2)
    print_vec('出力1', y)
    print(f'出力合計: {np.sum(y)}')

    return y, z1
```

In [44]:
```
x = np.array([1., 2., 3.])  # 入力値（入力層3に合わせる）
network =  init_network()
```

```
###ネットワーク初期化###
重み1：
[[0.48866237 0.2601217  0.98621247 0.3946938  0.85417186]
 [0.01870877 0.16358284 0.45155347 0.31676159 0.60743632]
 [0.4842468  0.97196884 0.32056517 0.19627568 0.50720291]]
W1.shape: (3, 5)
重み2：
[[0.11257419 0.19412603 0.72195902 0.45250326]
 [0.30615999 0.11289472 0.47140303 0.31015862]
 [0.14160989 0.94935021 0.78173798 0.95105029]
 [0.37689755 0.00649657 0.90473946 0.85787944]
 [0.48311459 0.55989536 0.48404571 0.07239544]]
W2.shape: (5, 4)
バイアス1：
[[0.55932185 0.37801083 0.76411616 0.18978849 0.74555344]]
b1.shape: (1, 5)
バイアス2：
[[0.58643475 0.32021856 0.12660376 0.87348712]]
b2.shape: (1, 4)
```

In [45]:
```
# 順伝搬による出力
y, z1 = forward(network, x)
```

```
###順伝搬を開始###
総入力1：
[[2.53814216 3.88120475 3.61513108 1.80683252 4.33620667]]
中間層出力1：
[[2.53814216 3.88120475 3.61513108 1.80683252 4.33620667]]
総入力2：
[[ 5.3482474   7.12269123 10.34837025  8.52793131]]
出力1：
[[0.00557514 0.03287663 0.82752647 0.13402176]]
出力合計: 1.0
```

回帰（3-5-4ネットワーク）

試してみよう ノードの構成を 3-5-4 に変更してみよう

In [46]:
```
def init_network():
    print('###ネットワーク初期化###')

    network = {}

    network['W1'] = np.random.rand(3, 5)
    network['W2'] = np.random.rand(5, 4)
    network['b1'] = np.random.rand(1, 5)
    network['b2'] = np.random.rand(1, 4)

    print_vec('重み1', network['W1'] )
    print('W1.shape: {}'.format(network['W1'].shape))
    print_vec('重み2', network['W2'] )
    print('W2.shape: {}'.format(network['W2'].shape))
    print_vec('バイアス1', network['b1'] )
    print('b1.shape: {}'.format(network['b1'].shape))
    print_vec('バイアス2', network['b2'] )
    print('b2.shape: {}'.format(network['b2'].shape))

    return network
```

In [47]:
```
def forward(network, x):
    print('###順伝搬を開始###')
    W1, W2 = network['W1'], network['W2']
    b1, b2 = network['b1'], network['b2']

    # 中間層 総入力: u1, 総出力: z1
    u1 = np.dot(x, W1) + b1
    z1 = functions.relu(u1)

    # 出力層 総入力: u2, 総出力: y
    u2 = np.dot(z1, W2) + b2
    # そのまま出力
    y = u2

    print_vec('総入力1', u1)
    print_vec('中間層出力1', z1)
    print_vec('総入力2', u2)
    print_vec('出力1', y)
    print(f'出力合計: {np.sum(y)}')

    return y, z1
```

In [48]:
```
x = np.array([1., 2., 3.])  # 入力値
network = init_network()  # ネットワーク初期化
```

```
###ネットワーク初期化###
重み1：
[[0.50573337 0.34906934 0.4709519  0.58536854 0.70173852]
 [0.20372545 0.30980928 0.69081682 0.4230652  0.90535275]
 [0.52859923 0.11625091 0.6069408  0.429425   0.31180646]]
W1.shape: (3, 5)
重み2：
[[0.2909784  0.89084196 0.46103565 0.60831641]
 [0.13772832 0.38520931 0.70351782 0.05695491]
 [0.68161913 0.45684204 0.59339805 0.43019482]
 [0.01057357 0.61758387 0.96980318 0.70321118]
 [0.89766565 0.4901329  0.04553562 0.90203199]]
W2.shape: (5, 4)
バイアス1：
[[0.65321508 0.18253888 0.38832422 0.36599526 0.91200781]]
b1.shape: (1, 5)
バイアス2：
[[0.84237419 0.04084094 0.08400568 0.59507685]]
b2.shape: (1, 4)
```

In [49]:
```
y, z1 = forward(network, x)
```

```
###順伝搬を開始###
総入力1：
[[3.15219704 1.49997952 4.06173216 3.08576919 4.35987121]]
中間層出力1：
[[3.15219704 1.49997952 4.06173216 3.08576919 4.35987121]]
総入力2：
[[ 8.68107365  9.324964    8.19388538 10.44806805]]
出力1：
[[ 8.68107365  9.324964    8.19388538 10.44806805]]
出力合計: 36.64799106825156
```

2値分類（5-10-1ネットワーク）

試してみよう_ノードの構成を 5-10-1 に変更してみよう

In [50]:
```
def init_network():
    print('###ネットワーク初期化###')

    network = {}

    network['W1'] = np.random.rand(5, 10)
    network['W2'] = np.random.rand(10, 1)
    network['b1'] = np.random.rand(1, 10)
    network['b2'] = np.random.rand(1, 1)

    print_vec('重み1', network['W1'] )
    print('W1.shape: {}'.format(network['W1'].shape))
    print_vec('重み2', network['W2'] )
    print('W2.shape: {}'.format(network['W2'].shape))
    print_vec('バイアス1', network['b1'] )
    print('b1.shape: {}'.format(network['b1'].shape))
    print_vec('バイアス2', network['b2'] )
    print('b2.shape: {}'.format(network['b2'].shape))

    return network
```

In [51]:
```
def forward(network, x):
    print('###順伝搬を開始###')
    W1, W2 = network['W1'], network['W2']
    b1, b2 = network['b1'], network['b2']

    # 中間層 総入力: u1, 総出力: z1
    u1 = np.dot(x, W1) + b1
    z1 = functions.relu(u1)

    # 出力層 総入力: u2, 総出力: y
    u2 = np.dot(z1, W2) + b2
    # sigmoid関数で結果出力
    y = functions.sigmoid(u2)

    print_vec('総入力1', u1)
    print_vec('中間層出力1', z1)
    print_vec('総入力2', u2)
    print_vec('出力1', y)
    print(f'出力合計: {np.sum(y)}')

    return y, z1
```

In [52]:
```
x = np.array([1., 2., 3., 4., 5.])  # 入力値
network = init_network()  # ネットワーク初期化
```

```
###ネットワーク初期化###
重み1：
[[0.80556972 0.77790036 0.90298165 0.33238872 0.38871919 0.64604122
  0.32525103 0.46014292 0.02188274 0.27914045]
 [0.74404004 0.62532999 0.21782424 0.90332755 0.3618047  0.24007042
  0.2272126  0.42513354 0.31091926 0.37818115]
 [0.06047842 0.02294171 0.27753144 0.35880716 0.2590297  0.72477
  0.31794083 0.32814934 0.83960394 0.01619333]
 [0.63977981 0.47808457 0.55020195 0.65991647 0.79317097 0.25882008
  0.13458216 0.63949856 0.86113943 0.9224108 ]
 [0.03145881 0.43477282 0.42416103 0.34961237 0.1854396  0.67302888
  0.98298669 0.84168774 0.44822376 0.73399109]]
W1.shape: (5, 10)
重み2：
[[0.09090956]
 [0.02760291]
 [0.76350037]
 [0.53089333]
 [0.18281407]
 [0.87995318]
 [0.6116286 ]
 [0.49423959]
 [0.04010926]
 [0.02160401]]
W2.shape: (10, 1)
バイアス1：
[[0.89156973 0.30539743 0.75099924 0.76858127 0.99342172 0.19089629
  0.88361951 0.22455919 0.56228825 0.23607931]]
b1.shape: (1, 10)
バイアス2：
[[0.21396118]]
b2.shape: (1, 1)
```

In [53]:
```
y, z1 = forward(network, x)
```

```
###順伝搬を開始###
総入力1：
[[6.08306811 6.48898527 7.24383664 8.37177428 6.98272133 7.89181308
  8.07038032 9.2858502  9.41049783 8.67976067]]
中間層出力1：
[[6.08306811 6.48898527 7.24383664 8.37177428 6.98272133 7.89181308
  8.07038032 9.2858502  9.41049783 8.67976067]]
総入力2：
[[29.23271784]]
出力1：
[[1.]]
出力合計: 0.9999999999997984
```

In [ ]:
```

```