simple RNN

バイナリ加算

In [1]:

```
import sys, os
sys.path.append(os.pardir)  # 親ディレクトリのファイルをインポートするための設定
import numpy as np
from common import functions
import matplotlib.pyplot as plt

# def d_tanh(x):



# データを用意
# 2進数の桁数
binary_dim = 8
# 最大値 + 1
largest_number = pow(2, binary_dim)
print("largest_number=",largest_number,"¥n")

# largest_numberまで2進数を用意(0～255)
binary = np.unpackbits(np.array([range(largest_number)],dtype=np.uint8).T,axis=1)
print("2進数 0～255")
print("binary=",binary)

input_layer_size = 2
hidden_layer_size = 16
output_layer_size = 1

weight_init_std = 1
learning_rate = 0.1

iters_num = 10000
plot_interval = 100

# ウェイト初期化 (バイアスは簡単のため省略)
W_in = weight_init_std * np.random.randn(input_layer_size, hidden_layer_size)
W_out = weight_init_std * np.random.randn(hidden_layer_size, output_layer_size)
W = weight_init_std * np.random.randn(hidden_layer_size, hidden_layer_size)

print("W_in",(input_layer_size, hidden_layer_size))
print(W_in)
print("W_out",(hidden_layer_size, output_layer_size))
print(W_out)
print("W",(hidden_layer_size, hidden_layer_size))
print(W)

# Xavier

# He

# 勾配
W_in_grad = np.zeros_like(W_in)
W_out_grad = np.zeros_like(W_out)
W_grad = np.zeros_like(W)
```

```
largest_number= 256

2進数 0～255
binary= [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 1]
 [0 0 0 ... 0 1 0]
 ...
 [1 1 1 ... 1 0 1]
 [1 1 1 ... 1 1 0]
 [1 1 1 ... 1 1 1]]
W_in (2, 16)
[[-0.87358767 -1.45052224  0.69814213 -0.63390555 -0.41601543  1.21406964
  -1.01670112  0.10338507 -0.36934196  0.20701404  0.17477337  0.2423453
   0.49770289 -0.73632334  0.66185044  0.29493996]
 [-0.0616437   0.1845211   0.39492862 -0.61129499 -0.37095096 -1.64456348
   1.28832409 -0.4849578   1.24737011 -1.13497967  0.01369891 -0.63075686
  -1.21485792  0.81588111  0.80777162 -0.30791495]]
W_out (16, 1)
[[-0.01329425]
 [-0.18465484]
 [ 1.43047087]
 [ 0.29346292]
 [-0.88706682]
 [-0.7443798 ]
 [-0.68491551]
 [-1.14634806]
 [-0.03708289]
 [-0.24231598]
 [ 1.5644554 ]
 [-0.72119509]
 [-0.39445817]
 [-1.85361671]
 [ 1.10341181]
 [ 0.1187674 ]]
W (16, 16)
[[-0.30801344  0.89218857  0.96187812  0.03619428  0.35543488 -0.23621656
   0.93838056 -0.96941035 -0.66005657 -0.46484726 -2.12932748  0.23893956
   0.82127334 -0.3804139   0.97060457 -0.11971208]
 [-0.42989745 -1.38873611 -0.21998635  0.5910179   1.16855833 -0.24803199
   0.60950587 -1.05914635 -0.13461636 -0.56949661 -1.27058929 -0.1631929
  -0.28612279  1.04997075  0.39005545 -0.93090723]
 [-0.78548448  1.32898789  0.48431379  1.18476748 -1.20583965  2.38106899
   0.70758657  0.3340393   0.16541444 -0.16272449  0.09848515  0.59963695
   0.18093494 -0.82179634  0.46961721 -0.68783078]
 [-0.41676775 -0.29297691  0.69161199  1.07408811  0.30693836 -0.18036443
   0.67841371 -1.09186957 -0.60491889  0.1319878  -0.52006943 -0.41759192
   0.15482427 -0.02272624 -0.42524429 -1.57445865]
 [-0.48933073 -0.7432153   0.09399359 -1.89804915  0.65972094  0.44631147
   1.45813969 -1.40804442  0.35577364 -1.21215862  0.50322716  0.00415579
   0.13725537 -0.07342864 -0.14695045 -1.08727773]
 [-0.39890971 -0.71163117  0.38282724 -1.1490658   0.9611157   0.00806778
  -1.7554231   0.76346729  0.30859779 -0.06808884  0.7631857   1.70881917
   0.22049866  0.56751452 -0.12792724  1.44311384]
 [-0.50300692 -0.13703532  2.20081228 -0.55080104  0.40530931 -0.43592542
   0.35870231  1.43711921  1.42199555  0.14811229  0.910212    0.31563983
  -0.84272312  0.8690483   0.08624721  0.89802758]
 [-1.08550718  2.51293925 -0.91732102 -1.40131224 -0.66604912  0.73624425
   0.87686191  1.653763   -0.19403514 -0.03441501  0.73927776 -0.68172108
  -0.01667272 -1.32138959  0.15686467  0.17330925]
 [ 0.80781847  0.65395747 -0.12491291  0.63049781 -0.49096063 -0.3322682
  -1.86174096  0.47716221 -1.55127187 -0.13571253 -0.43636134 -0.27897777
```

```
 -1.25032219 -1.05014034 -0.5690511   0.82623101]
[-0.94489677 -0.98320089 -0.60472302  0.2543434  -1.41374343  1.21058597
  0.31243528 -1.01037183 -1.33027376 -2.30653501 -0.91813586 -1.13925466
  0.72913088  0.20420714 -0.6481222  -0.82880983]
[-0.70734392 -0.2790617  -1.09713372 -0.31706022  1.97085099 -0.29803745
 -0.35705028  1.01619518 -0.31813163 -0.13701051 -1.53517536  2.78246122
 -0.59647226 -0.28139061  0.67481495 -1.28928203]
[-2.0291371  -0.11125028  0.49614038 -1.33931626 -0.02081994 -0.61837492
  0.10549755 -1.34756534  1.35458261 -0.843007   -0.45601996  1.52816201
 -1.42616131  0.35442967  0.51540791  0.19012444]
[ 0.83846685 -0.4378181   0.30645537  1.60472208 -0.85628419  0.42255116
 -0.47678774 -0.20680023 -0.37961904  0.25012987  0.12816065 -0.91027774
  0.16144123 -1.46604985 -0.2157068  -1.03179843]
[ 0.65778102 -0.70077268 -1.55473663  0.39392948 -0.52218763 -1.64232717
 -1.77041742  0.64658698 -1.60211612 -0.66375966  1.32947531 -0.5498168
 -0.0074396  -0.08143259 -1.28829081 -1.16848265]
[ 1.36503724  1.42301591  0.90155303  1.38208958 -0.71026055 -0.79858085
  0.11370876  1.90763648  1.08605586 -0.59543195  0.36420846  0.1079224
 -0.55808849 -0.69162752 -0.20824925  0.17618545]
[-0.15130041  0.29062057  0.95088201 -0.08756066  0.82299873 -1.18406327
 -0.88725138  0.88868299  0.59878716 -0.6682981   0.34844248  1.95257297
  0.53024169  0.22957499 -0.17890025  0.97189818]]
```

In [2]:

```
u = np.zeros((hidden_layer_size, binary_dim + 1))
z = np.zeros((hidden_layer_size, binary_dim + 1))
y = np.zeros((output_layer_size, binary_dim))
delta_out = np.zeros((output_layer_size, binary_dim))
delta = np.zeros((hidden_layer_size, binary_dim + 1))
print("u=",(hidden_layer_size, binary_dim + 1),u)
print("z=",(hidden_layer_size, binary_dim + 1),z)
print("y=",(output_layer_size, binary_dim),y)
print("delta_out=",(output_layer_size, binary_dim),delta_out)
print("delta=",(hidden_layer_size, binary_dim + 1),delta)
```

```
u= (16, 9) [[0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]]
z= (16, 9) [[0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]]
y= (1, 8) [[0. 0. 0. 0. 0. 0. 0. 0.]]
delta_out= (1, 8) [[0. 0. 0. 0. 0. 0. 0. 0.]]
delta= (16, 9) [[0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

In [3]:

```
all_losses = []

for i in range(iters_num):

    # A, B初期化 (a + b = d)
    a_int = np.random.randint(largest_number/2)
    a_bin = binary[a_int] # binary encoding
    b_int = np.random.randint(largest_number/2)
    b_bin = binary[b_int] # binary encoding

    # 正解データ
    d_int = a_int + b_int
    d_bin = binary[d_int]

    # 出力バイナリ
    out_bin = np.zeros_like(d_bin)

    # 時系列全体の誤差
    all_loss = 0

    # 時系列ループ
    for t in range(binary_dim):
        # 入力値
        X = np.array([a_bin[ - t - 1], b_bin[ - t - 1]]).reshape(1, -1)
        # 時刻tにおける正解データ
        dd = np.array([d_bin[binary_dim - t - 1]])

        u[:,t+1] = np.dot(X, W_in) + np.dot(z[:,t].reshape(1, -1), W)
        z[:,t+1] = functions.sigmoid(u[:,t+1])

        y[:,t] = functions.sigmoid(np.dot(z[:,t+1].reshape(1, -1), W_out))


        #誤差
        loss = functions.mean_squared_error(dd, y[:,t])

        delta_out[:,t] = functions.d_mean_squared_error(dd, y[:,t]) * functions.d_sigmoid(y[:,t
])

        all_loss += loss

        out_bin[binary_dim - t - 1] = np.round(y[:,t])


    for t in range(binary_dim)[::-1]:

        X = np.array([a_bin[-t-1],b_bin[-t-1]]).reshape(1, -1)

        delta[:,t] = (np.dot(delta[:,t+1].T, W.T) + np.dot(delta_out[:,t].T, W_out.T)) * functio
ns.d_sigmoid(u[:,t+1])

        # 勾配更新
        W_out_grad += np.dot(z[:,t+1].reshape(-1,1), delta_out[:,t].reshape(-1,1))
        W_grad += np.dot(z[:,t].reshape(-1,1), delta[:,t].reshape(1,-1))
        W_in_grad += np.dot(X.T, delta[:,t].reshape(1,-1))

    # 勾配適用
    W_in -= learning_rate * W_in_grad
    W_out -= learning_rate * W_out_grad
```

```
    W -= learning_rate * W_grad

    W_in_grad *= 0
    W_out_grad *= 0
    W_grad *= 0


    if(i % plot_interval == 0):
        all_losses.append(all_loss)
        print("iters:" + str(i))
        print("Loss:" + str(all_loss))
        print("Pred:" + str(out_bin))
        print("True:" + str(d_bin))
        out_int = 0
        for index,x in enumerate(reversed(out_bin)):
            out_int += x * pow(2, index)
        print(str(a_int) + " + " + str(b_int) + " = " + str(out_int))
        print("------------")

lists = range(0, iters_num, plot_interval)
plt.plot(lists, all_losses, label="loss")
plt.ylim(0,2.0)
plt.show()
```

```
iters:0
Loss:1.02324762924596
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 0 0 1 1]
51 + 96 = 0
------------
iters:100
Loss:0.8591926817987185
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 1 1 0 0 0]
24 + 64 = 0
------------
iters:200
Loss:0.8195749159192333
Pred:[0 0 0 1 0 0 0 0]
True:[0 1 0 1 0 0 0 1]
9 + 72 = 16
------------
iters:300
Loss:1.0547411148500954
Pred:[1 1 1 1 0 0 1 0]
True:[0 1 1 1 1 1 0 0]
65 + 59 = 242
------------
iters:400
Loss:0.9888278921351487
Pred:[0 1 1 0 1 0 1 0]
True:[0 0 1 1 1 0 0 1]
36 + 21 = 106
------------
iters:500
Loss:1.1295609466305188
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 0 1 0 1 1 0]
21 + 65 = 255
------------
iters:600
Loss:1.047600759767819
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 1 0 1 1 1]
2 + 85 = 0
------------
iters:700
Loss:0.8973631839217424
Pred:[1 1 1 1 1 1 1 0]
True:[1 0 1 1 0 1 1 1]
121 + 62 = 254
------------
iters:800
Loss:0.947940153726242
Pred:[0 1 1 0 1 0 0 0]
True:[0 1 0 0 1 0 0 1]
21 + 52 = 104
------------
iters:900
Loss:1.1909286043168046
Pred:[0 0 0 0 0 0 1 0]
True:[1 0 1 1 1 1 0 0]
65 + 123 = 2
------------
iters:1000
```

```
Loss:0.9418430168961416
Pred:[0 0 1 1 0 0 1 0]
True:[0 1 0 1 1 0 1 0]
25 + 65 = 50
------------
iters:1100
Loss:0.8227040466030088
Pred:[0 1 1 1 1 1 1 1]
True:[0 0 0 1 1 1 1 1]
15 + 16 = 127
------------
iters:1200
Loss:0.725658947384856
Pred:[0 0 0 0 0 1 1 0]
True:[0 1 0 0 0 1 1 0]
3 + 67 = 6
------------
iters:1300
Loss:0.9879283019506079
Pred:[1 1 0 0 0 0 1 0]
True:[1 0 1 0 0 0 0 0]
125 + 35 = 194
------------
iters:1400
Loss:0.7411730377365687
Pred:[0 0 0 0 1 0 1 0]
True:[0 0 0 0 1 0 1 0]
10 + 0 = 10
------------
iters:1500
Loss:0.8635429302968246
Pred:[0 0 1 1 1 1 1 0]
True:[0 0 1 1 0 1 0 0]
23 + 29 = 62
------------
iters:1600
Loss:0.8903159633243044
Pred:[0 1 0 0 0 0 0 1]
True:[1 0 1 0 0 1 0 1]
52 + 113 = 65
------------
iters:1700
Loss:0.7429002465327144
Pred:[1 1 1 0 0 1 1 0]
True:[1 1 1 0 0 1 0 0]
121 + 107 = 230
------------
iters:1800
Loss:0.5866759467184177
Pred:[1 1 1 0 1 1 1 0]
True:[1 1 1 0 1 1 1 0]
113 + 125 = 238
------------
iters:1900
Loss:0.6445346364143844
Pred:[1 1 0 0 0 0 0 1]
True:[1 1 1 0 0 0 0 1]
109 + 116 = 193
------------
iters:2000
Loss:1.0152734785940054
```

```
Pred:[0 1 1 1 1 0 0 1]
True:[1 0 0 0 0 0 0 1]
102 + 27 = 121
------------
iters:2100
Loss:0.39997369400949256
Pred:[0 1 0 1 1 0 0 1]
True:[0 1 0 1 1 0 0 1]
32 + 57 = 89
------------
iters:2200
Loss:0.4459480756800978
Pred:[1 0 0 1 0 1 0 0]
True:[1 0 1 1 0 1 0 0]
81 + 99 = 148
------------
iters:2300
Loss:0.5497551893675929
Pred:[0 1 1 0 0 1 1 1]
True:[0 1 0 0 0 1 1 1]
45 + 26 = 103
------------
iters:2400
Loss:0.28829973724217034
Pred:[0 1 0 0 1 0 0 1]
True:[0 1 0 0 1 0 0 1]
29 + 44 = 73
------------
iters:2500
Loss:0.18967498909378455
Pred:[1 1 0 1 0 0 1 0]
True:[1 1 0 1 0 0 1 0]
98 + 112 = 210
------------
iters:2600
Loss:0.14412228725731802
Pred:[1 0 0 0 1 0 0 1]
True:[1 0 0 0 1 0 0 1]
34 + 103 = 137
------------
iters:2700
Loss:0.05126302801524315
Pred:[0 0 0 1 0 1 1 0]
True:[0 0 0 1 0 1 1 0]
5 + 17 = 22
------------
iters:2800
Loss:0.07557993157275585
Pred:[1 0 1 0 1 1 1 0]
True:[1 0 1 0 1 1 1 0]
53 + 121 = 174
------------
iters:2900
Loss:0.0444225866925873
Pred:[0 1 1 0 1 1 1 0]
True:[0 1 1 0 1 1 1 0]
33 + 77 = 110
------------
iters:3000
Loss:0.021680900670464364
Pred:[0 0 0 1 0 1 1 0]
```

```
True:[0 0 0 1 0 1 1 0]
17 + 5 = 22
------------
iters:3100
Loss:0.041008682951315815
Pred:[1 1 0 1 0 0 0 0]
True:[1 1 0 1 0 0 0 0]
101 + 107 = 208
------------
iters:3200
Loss:0.026058468385165606
Pred:[1 0 1 1 0 1 0 0]
True:[1 0 1 1 0 1 0 0]
85 + 95 = 180
------------
iters:3300
Loss:0.03225248887610997
Pred:[0 1 0 1 0 1 1 0]
True:[0 1 0 1 0 1 1 0]
76 + 10 = 86
------------
iters:3400
Loss:0.027335808826009406
Pred:[1 1 0 1 0 1 0 0]
True:[1 1 0 1 0 1 0 0]
116 + 96 = 212
------------
iters:3500
Loss:0.018390540252587198
Pred:[1 0 0 0 1 0 0 0]
True:[1 0 0 0 1 0 0 0]
31 + 105 = 136
------------
iters:3600
Loss:0.026734951197049626
Pred:[0 1 1 0 1 1 1 1]
True:[0 1 1 0 1 1 1 1]
2 + 109 = 111
------------
iters:3700
Loss:0.00793906108269114
Pred:[0 0 1 0 1 1 1 0]
True:[0 0 1 0 1 1 1 0]
9 + 37 = 46
------------
iters:3800
Loss:0.00752156383889913
Pred:[0 0 1 0 1 1 1 0]
True:[0 0 1 0 1 1 1 0]
15 + 31 = 46
------------
iters:3900
Loss:0.005632957358962804
Pred:[1 1 1 0 0 1 0 0]
True:[1 1 1 0 0 1 0 0]
113 + 115 = 228
------------
iters:4000
Loss:0.010122913003593982
Pred:[1 1 0 1 0 1 0 1]
True:[1 1 0 1 0 1 0 1]
```

```
102 + 111 = 213
------------
iters:4100
Loss:0.005334921887662512
Pred:[1 1 1 0 1 0 1 0]
True:[1 1 1 0 1 0 1 0]
123 + 111 = 234
------------
iters:4200
Loss:0.010882294819784503
Pred:[1 0 0 1 0 0 0 1]
True:[1 0 0 1 0 0 0 1]
62 + 83 = 145
------------
iters:4300
Loss:0.011914910179920817
Pred:[1 0 1 1 0 1 1 1]
True:[1 0 1 1 0 1 1 1]
102 + 81 = 183
------------
iters:4400
Loss:0.013562092193149829
Pred:[1 0 0 1 1 0 1 0]
True:[1 0 0 1 1 0 1 0]
106 + 48 = 154
------------
iters:4500
Loss:0.010477783037175113
Pred:[1 1 0 0 0 1 0 0]
True:[1 1 0 0 0 1 0 0]
100 + 96 = 196
------------
iters:4600
Loss:0.0030531731736538545
Pred:[0 1 1 1 1 0 0 0]
True:[0 1 1 1 1 0 0 0]
73 + 47 = 120
------------
iters:4700
Loss:0.0062702169999958625
Pred:[0 1 0 0 1 1 0 1]
True:[0 1 0 0 1 1 0 1]
36 + 41 = 77
------------
iters:4800
Loss:0.0034973320247901458
Pred:[0 1 1 0 1 1 0 0]
True:[0 1 1 0 1 1 0 0]
27 + 81 = 108
------------
iters:4900
Loss:0.0061112306424434
Pred:[0 1 1 1 0 1 0 1]
True:[0 1 1 1 0 1 0 1]
58 + 59 = 117
------------
iters:5000
Loss:0.008749138350081762
Pred:[0 1 0 1 0 1 0 0]
True:[0 1 0 1 0 1 0 0]
76 + 8 = 84
```

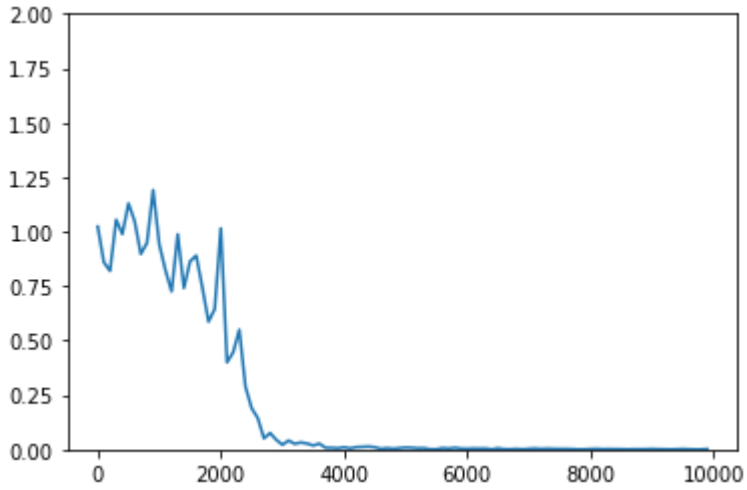
```
------------
iters:5100
Loss:0.007711474758783789
Pred:[1 0 0 0 0 1 0 0]
True:[1 0 0 0 0 1 0 0]
20 + 112 = 132
------------
iters:5200
Loss:0.005590559205246105
Pred:[0 1 1 1 1 1 1 1]
True:[0 1 1 1 1 1 1 1]
103 + 24 = 127
------------
iters:5300
Loss:0.005948569674817407
Pred:[1 0 0 1 0 1 0 1]
True:[1 0 0 1 0 1 0 1]
123 + 26 = 149
------------
iters:5400
Loss:0.0011359027871752272
Pred:[0 0 1 0 0 1 1 0]
True:[0 0 1 0 0 1 1 0]
19 + 19 = 38
------------
iters:5500
Loss:0.0007947308638731157
Pred:[1 1 0 0 1 0 1 0]
True:[1 1 0 0 1 0 1 0]
97 + 105 = 202
------------
iters:5600
Loss:0.00640856803948945
Pred:[1 0 1 1 1 0 1 1]
True:[1 0 1 1 1 0 1 1]
116 + 71 = 187
------------
iters:5700
Loss:0.004335212559065362
Pred:[1 0 0 1 1 1 1 1]
True:[1 0 0 1 1 1 1 1]
75 + 84 = 159
------------
iters:5800
Loss:0.007546358526511525
Pred:[0 1 0 1 0 0 0 0]
True:[0 1 0 1 0 0 0 0]
58 + 22 = 80
------------
iters:5900
Loss:0.004133972487325551
Pred:[1 0 1 0 1 1 1 1]
True:[1 0 1 0 1 1 1 1]
102 + 73 = 175
------------
iters:6000
Loss:0.0031931619971765535
Pred:[1 0 0 0 1 1 0 1]
True:[1 0 0 0 1 1 0 1]
73 + 68 = 141
------------
```

```
iters:6100
Loss:0.00450627748934388
Pred:[0 1 0 1 0 1 0 1]
True:[0 1 0 1 0 1 0 1]
32 + 53 = 85
------------
iters:6200
Loss:0.004053515949703609
Pred:[0 1 0 1 0 0 1 1]
True:[0 1 0 1 0 0 1 1]
21 + 62 = 83
------------
iters:6300
Loss:0.004433196590738335
Pred:[0 1 1 0 1 0 1 1]
True:[0 1 1 0 1 0 1 1]
47 + 60 = 107
------------
iters:6400
Loss:0.0007315915828292272
Pred:[0 0 1 1 1 0 1 0]
True:[0 0 1 1 1 0 1 0]
27 + 31 = 58
------------
iters:6500
Loss:0.005550585786678801
Pred:[1 1 1 1 0 1 0 0]
True:[1 1 1 1 0 1 0 0]
122 + 122 = 244
------------
iters:6600
Loss:0.0014685463031150368
Pred:[0 0 1 0 0 0 0 0]
True:[0 0 1 0 0 0 0 0]
7 + 25 = 32
------------
iters:6700
Loss:0.0011366471634662338
Pred:[1 0 0 0 1 1 0 0]
True:[1 0 0 0 1 1 0 0]
115 + 25 = 140
------------
iters:6800
Loss:0.0025944844385020127
Pred:[0 0 1 1 1 1 0 1]
True:[0 0 1 1 1 1 0 1]
9 + 52 = 61
------------
iters:6900
Loss:0.0009848134470022798
Pred:[0 1 1 0 0 0 0 0]
True:[0 1 1 0 0 0 0 0]
5 + 91 = 96
------------
iters:7000
Loss:0.002887893406051723
Pred:[0 0 1 0 1 1 1 1]
True:[0 0 1 0 1 1 1 1]
45 + 2 = 47
------------
iters:7100
```

```
Loss:0.004703533343046652
Pred:[0 1 0 1 0 1 0 0]
True:[0 1 0 1 0 1 0 0]
44 + 40 = 84
------------
iters:7200
Loss:0.002626317330035194
Pred:[1 0 1 1 1 0 0 1]
True:[1 0 1 1 1 0 0 1]
119 + 66 = 185
------------
iters:7300
Loss:0.003890765943095494
Pred:[0 0 1 1 1 0 0 0]
True:[0 0 1 1 1 0 0 0]
38 + 18 = 56
------------
iters:7400
Loss:0.003033246293940575
Pred:[1 0 0 1 1 1 1 1]
True:[1 0 0 1 1 1 1 1]
57 + 102 = 159
------------
iters:7500
Loss:0.0024020907006836504
Pred:[1 0 1 1 0 0 1 1]
True:[1 0 1 1 0 0 1 1]
55 + 124 = 179
------------
iters:7600
Loss:0.0027627530550140655
Pred:[1 0 0 0 1 1 0 1]
True:[1 0 0 0 1 1 0 1]
98 + 43 = 141
------------
iters:7700
Loss:0.001985165790826698
Pred:[0 1 1 1 0 1 0 1]
True:[0 1 1 1 0 1 0 1]
50 + 67 = 117
------------
iters:7800
Loss:0.0006182812161600005
Pred:[1 1 0 1 1 0 0 0]
True:[1 1 0 1 1 0 0 0]
95 + 121 = 216
------------
iters:7900
Loss:0.0007019586856717844
Pred:[1 0 0 0 1 0 1 0]
True:[1 0 0 0 1 0 1 0]
15 + 123 = 138
------------
iters:8000
Loss:0.0021045213446268243
Pred:[1 0 0 1 1 0 1 1]
True:[1 0 0 1 1 0 1 1]
66 + 89 = 155
------------
iters:8100
Loss:0.0031132510876862914
```

```
Pred:[0 1 0 0 1 0 0 0]
True:[0 1 0 0 1 0 0 0]
48 + 24 = 72
------------
iters:8200
Loss:0.001655879640384463
Pred:[1 1 1 0 0 0 0 1]
True:[1 1 1 0 0 0 0 1]
101 + 124 = 225
------------
iters:8300
Loss:0.0020724555663647302
Pred:[0 1 1 1 1 1 0 1]
True:[0 1 1 1 1 1 0 1]
90 + 35 = 125
------------
iters:8400
Loss:0.001776071243067434
Pred:[0 1 1 1 1 1 0 1]
True:[0 1 1 1 1 1 0 1]
82 + 43 = 125
------------
iters:8500
Loss:0.0014403478053908096
Pred:[0 1 1 1 0 0 1 1]
True:[0 1 1 1 0 0 1 1]
70 + 45 = 115
------------
iters:8600
Loss:0.0006595928035341434
Pred:[1 0 1 0 0 1 1 0]
True:[1 0 1 0 0 1 1 0]
119 + 47 = 166
------------
iters:8700
Loss:0.0014030552426216626
Pred:[1 0 1 0 0 0 0 1]
True:[1 0 1 0 0 0 0 1]
123 + 38 = 161
------------
iters:8800
Loss:0.0013815925052865143
Pred:[0 0 1 1 1 1 1 1]
True:[0 0 1 1 1 1 1 1]
48 + 15 = 63
------------
iters:8900
Loss:0.0014358685090234183
Pred:[1 1 0 0 0 0 0 1]
True:[1 1 0 0 0 0 0 1]
98 + 95 = 193
------------
iters:9000
Loss:0.0026526953578845545
Pred:[0 1 1 0 1 1 1 0]
True:[0 1 1 0 1 1 1 0]
54 + 56 = 110
------------
iters:9100
Loss:0.0015726959719423311
Pred:[1 0 1 1 1 0 1 1]
```

```
True:[1 0 1 1 1 0 1 1]
64 + 123 = 187
------------
iters:9200
Loss:0.0011584779588236188
Pred:[1 0 1 1 1 0 0 1]
True:[1 0 1 1 1 0 0 1]
63 + 122 = 185
------------
iters:9300
Loss:0.0004069756029068516
Pred:[1 0 1 0 0 0 0 0]
True:[1 0 1 0 0 0 0 0]
101 + 59 = 160
------------
iters:9400
Loss:0.0012055794325996046
Pred:[1 0 1 1 1 1 0 1]
True:[1 0 1 1 1 1 0 1]
119 + 70 = 189
------------
iters:9500
Loss:0.0024069022612522033
Pred:[1 0 0 0 1 0 0 0]
True:[1 0 0 0 1 0 0 0]
50 + 86 = 136
------------
iters:9600
Loss:0.0013478436898486665
Pred:[1 0 1 1 0 0 1 1]
True:[1 0 1 1 0 0 1 1]
61 + 118 = 179
------------
iters:9700
Loss:0.00025852276165899414
Pred:[0 1 1 0 1 0 1 0]
True:[0 1 1 0 1 0 1 0]
101 + 5 = 106
------------
iters:9800
Loss:0.0004493294721893731
Pred:[1 1 1 0 0 1 1 0]
True:[1 1 1 0 0 1 1 0]
103 + 127 = 230
------------
iters:9900
Loss:0.0021089196270958717
Pred:[0 1 0 1 0 0 1 0]
True:[0 1 0 1 0 0 1 0]
14 + 68 = 82
------------
```



[try] weight_init_stdを変更してみよう

In [4]:

```
# データを用意
# 2進数の桁数
binary_dim = 8
# 最大値 + 1
largest_number = pow(2, binary_dim)
# largest_numberまで2進数を用意
binary = np.unpackbits(np.array([range(largest_number)],dtype=np.uint8).T,axis=1)

input_layer_size = 2
hidden_layer_size = 16
output_layer_size = 1

weight_init_std = 100
learning_rate = 0.1

iters_num = 10000
plot_interval = 100

# ウェイト初期化 （バイアスは簡単のため省略）
W_in = weight_init_std * np.random.randn(input_layer_size, hidden_layer_size)
W_out = weight_init_std * np.random.randn(hidden_layer_size, output_layer_size)
W = weight_init_std * np.random.randn(hidden_layer_size, hidden_layer_size)

# Xavier


# He



# 勾配
W_in_grad = np.zeros_like(W_in)
W_out_grad = np.zeros_like(W_out)
W_grad = np.zeros_like(W)

u = np.zeros((hidden_layer_size, binary_dim + 1))
z = np.zeros((hidden_layer_size, binary_dim + 1))
y = np.zeros((output_layer_size, binary_dim))

delta_out = np.zeros((output_layer_size, binary_dim))
delta = np.zeros((hidden_layer_size, binary_dim + 1))

all_losses = []

for i in range(iters_num):

    # A, B初期化 (a + b = d)
    a_int = np.random.randint(largest_number/2)
    a_bin = binary[a_int] # binary encoding
    b_int = np.random.randint(largest_number/2)
    b_bin = binary[b_int] # binary encoding

    # 正解データ
    d_int = a_int + b_int
    d_bin = binary[d_int]

    # 出力バイナリ
    out_bin = np.zeros_like(d_bin)
```

```
    # 時系列全体の誤差
    all_loss = 0

    # 時系列ループ
    for t in range(binary_dim):
        # 入力値
        X = np.array([a_bin[ - t - 1], b_bin[ - t - 1]]).reshape(1, -1)
        # 時刻tにおける正解データ
        dd = np.array([d_bin[binary_dim - t - 1]])

        u[:,t+1] = np.dot(X, W_in) + np.dot(z[:,t].reshape(1, -1), W)
        z[:,t+1] = functions.sigmoid(u[:,t+1])

        y[:,t] = functions.sigmoid(np.dot(z[:,t+1].reshape(1, -1), W_out))


        #誤差
        loss = functions.mean_squared_error(dd, y[:,t])

        delta_out[:,t] = functions.d_mean_squared_error(dd, y[:,t]) * functions.d_sigmoid(y[:,t
])

        all_loss += loss

        out_bin[binary_dim - t - 1] = np.round(y[:,t])


    for t in range(binary_dim)[::-1]:
        X = np.array([a_bin[-t-1],b_bin[-t-1]]).reshape(1, -1)

        delta[:,t] = (np.dot(delta[:,t+1].T, W.T) + np.dot(delta_out[:,t].T, W_out.T)) * functio
ns.d_sigmoid(u[:,t+1])

        # 勾配更新
        W_out_grad += np.dot(z[:,t+1].reshape(-1,1), delta_out[:,t].reshape(-1,1))
        W_grad += np.dot(z[:,t].reshape(-1,1), delta[:,t].reshape(1,-1))
        W_in_grad += np.dot(X.T, delta[:,t].reshape(1,-1))

    # 勾配適用
    W_in -= learning_rate * W_in_grad
    W_out -= learning_rate * W_out_grad
    W -= learning_rate * W_grad

    W_in_grad *= 0
    W_out_grad *= 0
    W_grad *= 0


    if(i % plot_interval == 0):
        all_losses.append(all_loss)
        print("iters:" + str(i))
        print("Loss:" + str(all_loss))
        print("Pred:" + str(out_bin))
        print("True:" + str(d_bin))
        out_int = 0
        for index,x in enumerate(reversed(out_bin)):
            out_int += x * pow(2, index)
        print(str(a_int) + " + " + str(b_int) + " = " + str(out_int))
        print("------------")

lists = range(0, iters_num, plot_interval)
```

```
plt.plot(lists, all_losses, label="loss")
plt.ylim(0,2.0)
plt.show()
```

```
iters:0
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 1 1 0 0]
100 + 8 = 0
------------
iters:100
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 1 0 1 1 1 0]
11 + 35 = 0
------------
iters:200
Loss:3.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 1 1 1 1]
88 + 71 = 0
------------
iters:300
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 0 0 1 1 1]
92 + 75 = 0
------------
iters:400
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 1 1 0 0 0 1]
14 + 35 = 0
------------
iters:500
Loss:2.0
Pred:[1 0 0 0 0 0 0 1]
True:[1 0 0 1 1 1 1 1]
47 + 112 = 129
------------
iters:600
Loss:2.4874538011601697
Pred:[0 0 1 0 0 0 1 1]
True:[0 1 1 1 1 1 1 0]
80 + 46 = 35
------------
iters:700
Loss:1.5012058994535185
Pred:[1 0 0 0 0 1 0 0]
True:[1 1 0 0 0 0 1 0]
71 + 123 = 132
------------
iters:800
Loss:2.999994283603553
Pred:[0 0 0 0 0 0 1 1]
True:[1 1 0 1 0 1 0 0]
120 + 92 = 3
------------
iters:900
Loss:2.2923215817267817
Pred:[1 0 1 0 0 0 0 1]
True:[0 1 1 0 1 0 1 0]
53 + 53 = 161
------------
iters:1000
```

```
Loss:2.6924882255235767
Pred:[0 1 1 0 1 0 1 0]
True:[0 0 0 1 1 1 0 1]
4 + 25 = 106
------------
iters:1100
Loss:1.999999999999998
Pred:[0 0 0 0 0 1 0 1]
True:[0 1 1 1 0 0 0 1]
39 + 74 = 5
------------
iters:1200
Loss:2.521861217405468
Pred:[0 0 1 0 0 1 1 1]
True:[0 1 1 0 1 0 0 0]
30 + 74 = 39
------------
iters:1300
Loss:1.9998156341710962
Pred:[0 1 1 1 0 1 0 1]
True:[1 0 0 0 0 1 0 1]
95 + 38 = 117
------------
iters:1400
Loss:3.4999998225689084
Pred:[0 1 0 1 0 1 1 1]
True:[0 0 1 0 1 0 0 0]
14 + 26 = 87
------------
iters:1500
Loss:2.3425392667017615
Pred:[0 1 0 0 0 0 1 1]
True:[0 1 1 1 0 1 0 0]
106 + 10 = 67
------------
iters:1600
Loss:2.59342157549972
Pred:[1 0 1 0 0 1 1 1]
True:[0 1 0 0 1 0 1 1]
63 + 12 = 167
------------
iters:1700
Loss:2.972543682198527
Pred:[0 1 1 1 0 0 1 1]
True:[1 0 0 0 0 0 0 0]
89 + 39 = 115
------------
iters:1800
Loss:0.9927458019526363
Pred:[1 0 1 1 1 1 0 1]
True:[0 0 1 1 1 1 0 0]
51 + 9 = 189
------------
iters:1900
Loss:2.0000000039693373
Pred:[0 1 1 1 0 1 1 1]
True:[0 1 1 1 1 0 0 0]
78 + 42 = 119
------------
iters:2000
Loss:3.1229445812606724
```

```
Pred:[1 0 1 0 0 0 1 1]
True:[0 1 0 1 1 1 1 0]
94 + 0 = 163
------------
iters:2100
Loss:1.4165414369907974
Pred:[0 1 0 1 1 1 0 1]
True:[0 1 1 1 1 1 1 0]
11 + 115 = 93
------------
iters:2200
Loss:1.975375495713911
Pred:[1 0 0 0 0 0 1 0]
True:[1 1 0 1 0 0 0 1]
88 + 121 = 130
------------
iters:2300
Loss:2.5389275456880456
Pred:[0 0 0 0 1 0 0 1]
True:[0 1 1 0 0 0 1 0]
9 + 89 = 9
------------
iters:2400
Loss:2.4957913616074556
Pred:[1 1 0 0 0 1 0 1]
True:[1 0 1 1 0 1 1 0]
59 + 123 = 197
------------
iters:2500
Loss:1.0139500151358865
Pred:[0 1 1 1 1 1 1 1]
True:[0 1 1 0 0 1 1 1]
7 + 96 = 127
------------
iters:2600
Loss:3.334312728670006
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 1 1 0 1 1 1]
122 + 125 = 0
------------
iters:2700
Loss:2.7033201498864874
Pred:[1 1 1 0 0 0 1 0]
True:[0 0 0 1 0 0 0 1]
8 + 9 = 226
------------
iters:2800
Loss:2.314514272289334
Pred:[0 1 1 0 1 1 0 0]
True:[0 0 0 0 1 0 1 1]
2 + 9 = 108
------------
iters:2900
Loss:0.9918959490463053
Pred:[1 1 0 0 0 0 0 1]
True:[1 1 0 0 1 0 0 0]
107 + 93 = 193
------------
iters:3000
Loss:1.9759264728272536
Pred:[0 0 0 1 1 1 0 1]
```

```
True:[1 0 1 1 1 1 1 0]
69 + 121 = 29
------------
iters:3100
Loss:2.4944124024577685
Pred:[1 1 0 0 0 0 0 0]
True:[1 0 0 1 0 1 1 1]
122 + 29 = 192
------------
iters:3200
Loss:2.4999907235539065
Pred:[0 0 0 0 0 0 1 1]
True:[0 1 1 1 1 0 1 0]
12 + 110 = 3
------------
iters:3300
Loss:1.1703067548161932
Pred:[0 1 0 1 1 1 0 1]
True:[0 1 1 1 1 1 1 0]
11 + 115 = 93
------------
iters:3400
Loss:1.9999999980527587
Pred:[1 1 1 0 1 1 1 1]
True:[0 0 0 1 1 1 1 1]
31 + 0 = 239
------------
iters:3500
Loss:1.980314968736584
Pred:[1 1 0 1 0 1 1 1]
True:[0 1 0 0 0 0 0 1]
53 + 12 = 215
------------
iters:3600
Loss:3.141962917200856
Pred:[1 1 0 0 0 1 0 0]
True:[0 1 1 1 1 0 1 1]
118 + 5 = 196
------------
iters:3700
Loss:2.0027157127375026
Pred:[1 0 1 0 1 1 0 1]
True:[1 0 0 1 0 1 1 0]
115 + 35 = 173
------------
iters:3800
Loss:1.9937605361687138
Pred:[0 1 0 0 0 0 0 0]
True:[0 1 1 0 0 1 1 1]
2 + 101 = 64
------------
iters:3900
Loss:2.5266957630114493
Pred:[0 1 1 1 0 0 0 1]
True:[0 0 0 0 1 1 1 1]
9 + 6 = 113
------------
iters:4000
Loss:2.0441435614125623
Pred:[0 1 1 1 1 1 0 1]
True:[1 0 1 0 0 1 0 1]
```

```
123 + 42 = 125
------------
iters:4100
Loss:2.9876304737633186
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 1 1 1 1 1 1]
2 + 61 = 0
------------
iters:4200
Loss:1.0479105275825369
Pred:[1 0 1 0 1 1 1 0]
True:[1 0 0 0 1 1 1 1]
92 + 51 = 174
------------
iters:4300
Loss:1.3325047503712402
Pred:[1 0 0 0 0 0 1 0]
True:[1 0 0 1 1 0 1 1]
48 + 107 = 130
------------
iters:4400
Loss:3.7400752408246483
Pred:[0 1 0 1 1 1 1 1]
True:[1 0 1 0 0 0 0 0]
104 + 56 = 95
------------
iters:4500
Loss:1.8055474908393032
Pred:[1 1 1 1 0 1 0 0]
True:[1 0 0 1 1 1 0 1]
74 + 83 = 244
------------
iters:4600
Loss:1.1975114417531973
Pred:[1 1 0 0 0 1 0 1]
True:[1 0 0 0 0 0 0 1]
62 + 67 = 197
------------
iters:4700
Loss:1.7888703993916777
Pred:[0 0 1 1 1 0 1 1]
True:[1 0 1 1 0 1 1 0]
112 + 70 = 59
------------
iters:4800
Loss:2.499983498180411
Pred:[1 0 1 0 1 1 1 1]
True:[0 1 0 0 0 0 1 1]
51 + 16 = 175
------------
iters:4900
Loss:2.163038747785811
Pred:[1 1 1 1 0 0 0 1]
True:[0 0 0 1 0 1 1 0]
13 + 9 = 241
------------
iters:5000
Loss:0.6750044434535217
Pred:[1 0 1 0 0 1 0 1]
True:[1 0 1 0 1 1 0 1]
98 + 75 = 165
```

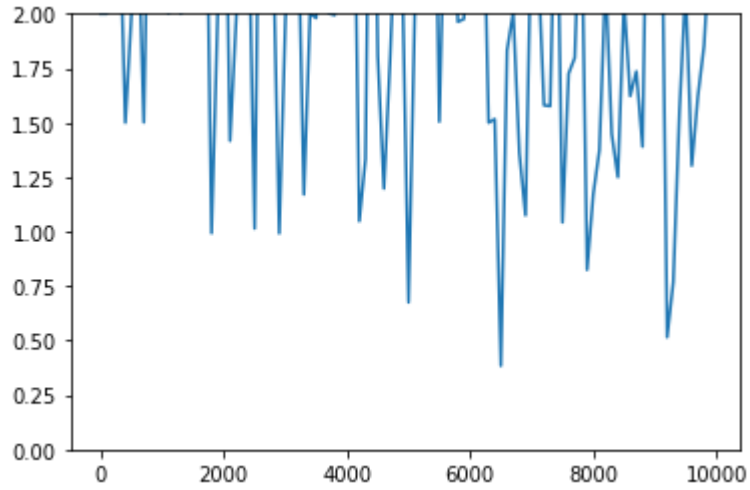
```
------------
iters:5100
Loss:1.9966016479156423
Pred:[0 1 0 0 1 0 0 1]
True:[0 1 0 1 0 1 0 0]
3 + 81 = 73
------------
iters:5200
Loss:2.4997907957768843
Pred:[0 1 1 0 0 0 0 1]
True:[1 1 0 1 1 0 0 0]
107 + 109 = 97
------------
iters:5300
Loss:3.6246776862363577
Pred:[0 1 1 0 0 0 1 1]
True:[1 0 0 1 1 1 0 0]
100 + 56 = 99
------------
iters:5400
Loss:3.2158084098544797
Pred:[1 0 0 1 0 0 1 1]
True:[0 1 1 0 1 1 1 0]
98 + 12 = 147
------------
iters:5500
Loss:1.5044531150371228
Pred:[1 0 1 0 0 1 1 1]
True:[1 1 1 0 1 0 1 1]
112 + 123 = 167
------------
iters:5600
Loss:2.679103142774528
Pred:[0 1 1 0 1 0 1 1]
True:[1 0 0 1 1 1 0 1]
68 + 89 = 107
------------
iters:5700
Loss:2.9999812517966173
Pred:[1 0 1 0 1 0 0 0]
True:[0 1 1 1 0 0 1 1]
30 + 85 = 168
------------
iters:5800
Loss:1.9628326720450044
Pred:[1 1 0 1 0 0 1 0]
True:[1 0 0 0 1 0 1 1]
60 + 79 = 210
------------
iters:5900
Loss:1.9762079997293642
Pred:[0 1 0 0 0 0 0 0]
True:[0 1 0 1 0 1 1 1]
74 + 13 = 64
------------
iters:6000
Loss:3.499999994221983
Pred:[1 0 1 1 1 0 1 1]
True:[0 1 1 0 0 1 0 0]
25 + 75 = 187
------------
```

```
iters:6100
Loss:2.995307419322123
Pred:[1 0 1 1 0 1 0 1]
True:[0 1 1 0 0 0 1 0]
27 + 71 = 181
------------
iters:6200
Loss:2.789718070464144
Pred:[1 1 0 1 0 1 0 1]
True:[0 1 1 0 1 1 1 0]
55 + 55 = 213
------------
iters:6300
Loss:1.4999999939443962
Pred:[0 1 1 1 1 0 1 1]
True:[0 0 1 1 1 0 0 0]
45 + 11 = 123
------------
iters:6400
Loss:1.5165615721014072
Pred:[1 1 0 1 0 1 0 1]
True:[1 1 0 1 1 1 1 0]
111 + 111 = 213
------------
iters:6500
Loss:0.3839356135345106
Pred:[0 1 0 1 0 1 0 0]
True:[0 1 0 1 0 1 0 1]
50 + 35 = 84
------------
iters:6600
Loss:1.8321620941498025
Pred:[0 1 1 0 1 1 1 1]
True:[0 0 0 1 1 1 1 0]
14 + 16 = 111
------------
iters:6700
Loss:2.0065658042231274
Pred:[0 1 0 1 0 0 1 1]
True:[1 1 1 0 0 0 1 0]
108 + 118 = 83
------------
iters:6800
Loss:1.3605595340021164
Pred:[1 0 0 1 1 0 0 0]
True:[1 0 1 0 1 0 0 1]
76 + 93 = 152
------------
iters:6900
Loss:1.0746384726336506
Pred:[0 1 1 0 1 0 1 1]
True:[0 1 1 0 1 1 1 0]
68 + 42 = 107
------------
iters:7000
Loss:2.4991982067975944
Pred:[0 1 1 1 1 0 1 1]
True:[0 1 1 0 0 1 0 0]
93 + 7 = 123
------------
iters:7100
```

```
Loss:2.2640700316834734
Pred:[1 0 1 0 1 1 1 1]
True:[0 1 0 1 1 0 1 1]
21 + 70 = 175
------------
iters:7200
Loss:1.5800776188375496
Pred:[0 1 1 0 1 0 0 1]
True:[0 1 1 1 1 0 1 0]
81 + 41 = 105
------------
iters:7300
Loss:1.5761398290534976
Pred:[1 0 1 1 0 0 1 1]
True:[1 0 0 0 0 0 1 0]
8 + 122 = 179
------------
iters:7400
Loss:2.8308128113063655
Pred:[1 1 0 1 1 0 1 1]
True:[1 0 1 0 1 1 0 0]
72 + 100 = 219
------------
iters:7500
Loss:1.0422466742893564
Pred:[0 1 1 0 0 0 1 1]
True:[0 1 1 0 1 0 1 0]
72 + 34 = 99
------------
iters:7600
Loss:1.7252890574797564
Pred:[1 1 0 1 1 1 0 0]
True:[1 1 1 0 0 1 0 1]
124 + 105 = 220
------------
iters:7700
Loss:1.7982955037157442
Pred:[0 1 1 0 1 0 1 1]
True:[0 0 1 1 0 1 1 0]
16 + 38 = 107
------------
iters:7800
Loss:2.640052594266052
Pred:[0 1 1 0 1 1 0 1]
True:[1 0 0 0 0 0 0 0]
87 + 41 = 109
------------
iters:7900
Loss:0.8244297428608508
Pred:[1 1 0 1 1 1 0 0]
True:[0 0 0 1 1 1 0 1]
26 + 3 = 220
------------
iters:8000
Loss:1.1711529890490877
Pred:[1 0 1 0 0 1 0 1]
True:[1 0 1 0 0 1 1 0]
78 + 88 = 165
------------
iters:8100
Loss:1.374064061047449
```

```
Pred:[1 0 1 1 1 1 1 0]
True:[1 0 0 0 1 1 1 1]
64 + 79 = 190
------------
iters:8200
Loss:2.1628155520328844
Pred:[1 0 1 0 1 1 0 1]
True:[0 1 1 0 0 0 0 0]
7 + 89 = 173
------------
iters:8300
Loss:1.4485179747576866
Pred:[0 1 1 0 1 1 0 0]
True:[0 1 1 0 1 0 1 1]
58 + 49 = 108
------------
iters:8400
Loss:1.2499833603261283
Pred:[1 0 1 1 0 1 1 1]
True:[0 1 1 1 1 1 1 1]
53 + 74 = 183
------------
iters:8500
Loss:2.059287142684212
Pred:[1 0 1 1 0 1 0 1]
True:[1 0 1 1 1 0 1 0]
70 + 116 = 181
------------
iters:8600
Loss:1.6223920326055525
Pred:[1 0 1 1 1 1 1 0]
True:[1 0 0 0 1 0 1 1]
20 + 119 = 190
------------
iters:8700
Loss:1.7354788019897491
Pred:[1 0 1 0 1 0 0 1]
True:[0 1 1 0 1 0 1 0]
21 + 85 = 169
------------
iters:8800
Loss:1.3898787188989534
Pred:[0 1 1 0 1 1 1 1]
True:[0 1 0 1 1 0 1 1]
45 + 46 = 111
------------
iters:8900
Loss:3.171081786689384
Pred:[1 0 1 0 1 1 0 1]
True:[0 1 0 1 0 0 0 0]
14 + 66 = 173
------------
iters:9000
Loss:2.4127970239168106
Pred:[0 0 1 0 1 0 0 1]
True:[1 1 0 0 1 0 1 0]
81 + 121 = 41
------------
iters:9100
Loss:2.819889350753921
Pred:[1 0 1 0 1 0 0 1]
```

```
True:[0 1 1 1 0 1 0 0]
43 + 73 = 169
------------
iters:9200
Loss:0.5151635318674928
Pred:[0 1 1 0 1 1 1 1]
True:[0 1 0 0 1 1 1 1]
35 + 44 = 111
------------
iters:9300
Loss:0.769350253773349
Pred:[1 1 0 0 1 1 1 0]
True:[1 0 0 0 1 1 1 1]
84 + 59 = 206
------------
iters:9400
Loss:1.551657062046964
Pred:[0 1 1 0 1 1 0 0]
True:[1 0 1 1 1 1 0 1]
106 + 83 = 108
------------
iters:9500
Loss:2.1207910202075673
Pred:[1 0 1 0 1 1 1 1]
True:[0 1 0 1 1 0 1 1]
71 + 20 = 175
------------
iters:9600
Loss:1.3026333301422626
Pred:[1 0 1 1 1 0 1 0]
True:[0 0 0 1 0 0 1 1]
4 + 15 = 186
------------
iters:9700
Loss:1.622217272900473
Pred:[1 0 0 1 0 0 0 1]
True:[0 1 1 1 0 0 0 0]
34 + 78 = 145
------------
iters:9800
Loss:1.8542940968146
Pred:[0 1 1 0 1 1 0 1]
True:[1 0 1 0 0 1 0 0]
51 + 113 = 109
------------
iters:9900
Loss:2.3441571913508037
Pred:[1 1 0 0 1 0 0 0]
True:[1 0 1 0 0 1 1 0]
66 + 100 = 200
------------
```



[try] learning_rateを変更してみよう

In [5]:

```
input_layer_size = 2
hidden_layer_size = 16
output_layer_size = 1

weight_init_std = 1
learning_rate = 10

iters_num = 10000
plot_interval = 100

# ウェイト初期化 (バイアスは簡単のため省略)
W_in = weight_init_std * np.random.randn(input_layer_size, hidden_layer_size)
W_out = weight_init_std * np.random.randn(hidden_layer_size, output_layer_size)
W = weight_init_std * np.random.randn(hidden_layer_size, hidden_layer_size)

# Xavier


# He



# 勾配
W_in_grad = np.zeros_like(W_in)
W_out_grad = np.zeros_like(W_out)
W_grad = np.zeros_like(W)

u = np.zeros((hidden_layer_size, binary_dim + 1))
z = np.zeros((hidden_layer_size, binary_dim + 1))
y = np.zeros((output_layer_size, binary_dim))

delta_out = np.zeros((output_layer_size, binary_dim))
delta = np.zeros((hidden_layer_size, binary_dim + 1))

all_losses = []

for i in range(iters_num):

    # A, B初期化 (a + b = d)
    a_int = np.random.randint(largest_number/2)
    a_bin = binary[a_int] # binary encoding
    b_int = np.random.randint(largest_number/2)
    b_bin = binary[b_int] # binary encoding

    # 正解データ
    d_int = a_int + b_int
    d_bin = binary[d_int]

    # 出力バイナリ
    out_bin = np.zeros_like(d_bin)

    # 時系列全体の誤差
    all_loss = 0

    # 時系列ループ
    for t in range(binary_dim):
        # 入力値
        X = np.array([a_bin[ - t - 1], b_bin[ - t - 1]]).reshape(1, -1)
        # 時刻tにおける正解データ
```

```
        dd = np.array([d_bin[binary_dim - t - 1]])

        u[:,t+1] = np.dot(X, W_in) + np.dot(z[:,t].reshape(1, -1), W)
        z[:,t+1] = functions.sigmoid(u[:,t+1])

        y[:,t] = functions.sigmoid(np.dot(z[:,t+1].reshape(1, -1), W_out))


        #誤差
        loss = functions.mean_squared_error(dd, y[:,t])

        delta_out[:,t] = functions.d_mean_squared_error(dd, y[:,t]) * functions.d_sigmoid(y[:,t
])

        all_loss += loss

        out_bin[binary_dim - t - 1] = np.round(y[:,t])


    for t in range(binary_dim)[::-1]:
        X = np.array([a_bin[-t-1],b_bin[-t-1]]).reshape(1, -1)

        delta[:,t] = (np.dot(delta[:,t+1].T, W.T) + np.dot(delta_out[:,t].T, W_out.T)) * functio
ns.d_sigmoid(u[:,t+1])

        # 勾配更新
        W_out_grad += np.dot(z[:,t+1].reshape(-1,1), delta_out[:,t].reshape(-1,1))
        W_grad += np.dot(z[:,t].reshape(-1,1), delta[:,t].reshape(1,-1))
        W_in_grad += np.dot(X.T, delta[:,t].reshape(1,-1))

    # 勾配適用
    W_in -= learning_rate * W_in_grad
    W_out -= learning_rate * W_out_grad
    W -= learning_rate * W_grad

    W_in_grad *= 0
    W_out_grad *= 0
    W_grad *= 0


    if(i % plot_interval == 0):
        all_losses.append(all_loss)
        print("iters:" + str(i))
        print("Loss:" + str(all_loss))
        print("Pred:" + str(out_bin))
        print("True:" + str(d_bin))
        out_int = 0
        for index,x in enumerate(reversed(out_bin)):
            out_int += x * pow(2, index)
        print(str(a_int) + " + " + str(b_int) + " = " + str(out_int))
        print("------------")

lists = range(0, iters_num, plot_interval)
plt.plot(lists, all_losses, label="loss")
plt.ylim(0,2.0)
plt.show()
```

```
iters:0
Loss:1.8550955788817987
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 0 1 0 1 0 0]
17 + 67 = 255
------------
iters:100
Loss:0.6905353718033687
Pred:[1 1 1 0 1 1 0 1]
True:[1 0 1 0 1 1 0 1]
97 + 76 = 237
------------
iters:200
Loss:0.7055038097039731
Pred:[0 1 0 0 0 0 0 1]
True:[0 1 1 1 0 0 1 1]
61 + 54 = 65
------------
iters:300
Loss:1.4889404881790158
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 1 0 0 1 1 1]
49 + 54 = 255
------------
iters:400
Loss:0.7770919781230006
Pred:[1 1 1 1 1 1 1 0]
True:[0 1 1 1 1 1 0 0]
14 + 110 = 254
------------
iters:500
Loss:1.4843493522788644
Pred:[0 1 1 1 1 1 1 1]
True:[0 1 0 1 1 0 0 1]
31 + 58 = 127
------------
iters:600
Loss:0.7833207291430755
Pred:[0 0 1 0 1 0 0 0]
True:[0 1 1 0 1 1 0 0]
34 + 74 = 40
------------
iters:700
Loss:0.8357092881262331
Pred:[0 1 1 0 1 1 0 0]
True:[0 1 0 1 1 1 0 0]
89 + 3 = 108
------------
iters:800
Loss:0.7500469628367811
Pred:[0 0 1 1 1 0 1 0]
True:[0 0 1 1 1 1 0 0]
34 + 26 = 58
------------
iters:900
Loss:1.1997909075749447
Pred:[0 0 0 0 0 1 0 0]
True:[1 0 1 0 1 1 1 0]
61 + 113 = 4
------------
iters:1000
```

```
Loss:0.8792026576352072
Pred:[0 1 0 1 1 1 1 1]
True:[0 1 1 0 1 0 0 0]
11 + 93 = 95
------------
iters:1100
Loss:1.1959250698942332
Pred:[0 0 1 1 1 0 0 1]
True:[0 1 0 0 1 0 0 1]
17 + 56 = 57
------------
iters:1200
Loss:0.9463220063845974
Pred:[0 1 1 0 1 1 1 0]
True:[1 0 0 0 1 1 0 0]
104 + 36 = 110
------------
iters:1300
Loss:1.3024790788874798
Pred:[1 0 0 0 0 0 0 1]
True:[0 0 1 0 0 1 1 1]
16 + 23 = 129
------------
iters:1400
Loss:0.6179273564300028
Pred:[0 0 0 0 0 0 0 1]
True:[0 1 0 0 0 1 0 1]
14 + 55 = 1
------------
iters:1500
Loss:0.8278684995140099
Pred:[0 1 1 1 1 1 1 1]
True:[1 0 0 1 1 1 1 1]
61 + 98 = 127
------------
iters:1600
Loss:0.5020427411069404
Pred:[0 0 1 1 1 1 1 0]
True:[1 0 0 1 0 1 1 0]
64 + 86 = 62
------------
iters:1700
Loss:1.1398836500078042
Pred:[1 0 0 0 0 1 0 0]
True:[1 0 1 1 1 1 0 0]
100 + 88 = 132
------------
iters:1800
Loss:0.9758164292669792
Pred:[0 1 0 1 0 0 1 1]
True:[0 1 0 0 1 1 1 1]
66 + 13 = 83
------------
iters:1900
Loss:0.19462257652104129
Pred:[1 0 0 1 0 0 0 1]
True:[1 0 0 1 0 0 0 1]
66 + 79 = 145
------------
iters:2000
Loss:0.4963471211279579
```

```
Pred:[0 1 0 0 0 1 0 0]
True:[0 1 1 0 0 1 1 0]
56 + 46 = 68
------------
iters:2100
Loss:1.1713854625831477
Pred:[1 1 1 1 1 0 1 0]
True:[1 0 0 0 0 0 1 0]
72 + 58 = 250
------------
iters:2200
Loss:0.9996692866006675
Pred:[0 0 0 1 0 0 0 0]
True:[0 0 0 1 1 1 0 0]
15 + 13 = 16
------------
iters:2300
Loss:0.6490851167352016
Pred:[1 1 1 1 1 1 0 0]
True:[0 1 1 0 1 1 0 0]
0 + 108 = 252
------------
iters:2400
Loss:0.8318103015544787
Pred:[1 0 1 0 1 1 1 0]
True:[1 0 0 1 1 1 1 0]
90 + 68 = 174
------------
iters:2500
Loss:0.47092262982110744
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 1 0 1 1 1 1]
4 + 107 = 255
------------
iters:2600
Loss:1.0832116486767676
Pred:[1 1 1 1 1 1 1 1]
True:[1 0 1 0 0 0 0 1]
103 + 58 = 255
------------
iters:2700
Loss:1.1040576340489734
Pred:[1 1 1 1 1 1 1 1]
True:[1 1 0 0 0 0 0 1]
98 + 95 = 255
------------
iters:2800
Loss:0.4569445865185857
Pred:[1 0 0 0 1 1 1 1]
True:[1 0 1 1 1 1 1 1]
110 + 81 = 143
------------
iters:2900
Loss:1.6802473456554352
Pred:[1 1 1 0 1 1 1 1]
True:[1 0 0 0 1 1 1 0]
45 + 97 = 239
------------
iters:3000
Loss:0.408909820111784
Pred:[1 0 0 0 1 1 0 0]
```

```
True:[1 0 0 1 1 1 0 0]
98 + 58 = 140
------------
iters:3100
Loss:0.3713880089016609
Pred:[0 1 0 0 0 0 0 1]
True:[0 1 0 0 1 0 0 1]
1 + 72 = 65
------------
iters:3200
Loss:1.4580734064202132
Pred:[1 1 1 1 1 1 1 0]
True:[0 1 1 1 0 0 1 0]
8 + 106 = 254
------------
iters:3300
Loss:0.9765015717555201
Pred:[1 1 1 1 1 1 1 0]
True:[0 1 1 1 1 1 0 0]
124 + 0 = 254
------------
iters:3400
Loss:2.4993743867016045
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 0 1 0 0 1 0]
11 + 71 = 255
------------
iters:3500
Loss:1.3965272742934587
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 1 1 0 1 0 0]
24 + 28 = 0
------------
iters:3600
Loss:1.3895191063291106
Pred:[1 1 1 1 1 1 0 1]
True:[0 0 1 0 1 0 0 1]
21 + 20 = 253
------------
iters:3700
Loss:2.286838717483974
Pred:[0 1 1 1 1 1 1 1]
True:[1 0 0 1 0 1 0 1]
95 + 54 = 127
------------
iters:3800
Loss:1.380238980272415
Pred:[0 0 0 0 0 0 0 1]
True:[1 0 0 0 1 1 1 1]
49 + 94 = 1
------------
iters:3900
Loss:0.9857566656953534
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 1 1 1 0 1 1]
89 + 34 = 255
------------
iters:4000
Loss:2.1340658569199684
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 1 0 1 1]
```

```
66 + 41 = 0
------------
iters:4100
Loss:1.2864288632251386
Pred:[1 1 0 1 1 1 1 0]
True:[0 1 1 1 0 1 0 0]
84 + 32 = 222
------------
iters:4200
Loss:1.499998965261156
Pred:[0 0 0 0 0 0 0 1]
True:[1 0 0 1 1 0 0 1]
38 + 115 = 1
------------
iters:4300
Loss:1.0785497477003212
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 1 1 0 0]
55 + 101 = 0
------------
iters:4400
Loss:1.8073812423044429
Pred:[1 0 1 0 1 0 1 1]
True:[1 1 0 0 0 0 0 1]
98 + 95 = 171
------------
iters:4500
Loss:0.9444027399536179
Pred:[0 0 0 0 0 0 0 1]
True:[1 0 1 0 0 0 0 1]
61 + 100 = 1
------------
iters:4600
Loss:1.4968774302256045
Pred:[1 1 1 1 1 1 1 1]
True:[1 0 1 0 0 1 1 1]
64 + 103 = 255
------------
iters:4700
Loss:1.9999768474627224
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 0 1 1 0]
104 + 46 = 0
------------
iters:4800
Loss:1.1992398403121027
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 1 0 1 0]
26 + 112 = 0
------------
iters:4900
Loss:1.4305378379925835
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 1 0 1 0 1 1]
0 + 107 = 255
------------
iters:5000
Loss:1.006220768025151
Pred:[0 0 0 1 1 1 1 1]
True:[0 0 0 1 1 0 0 1]
10 + 15 = 31
```

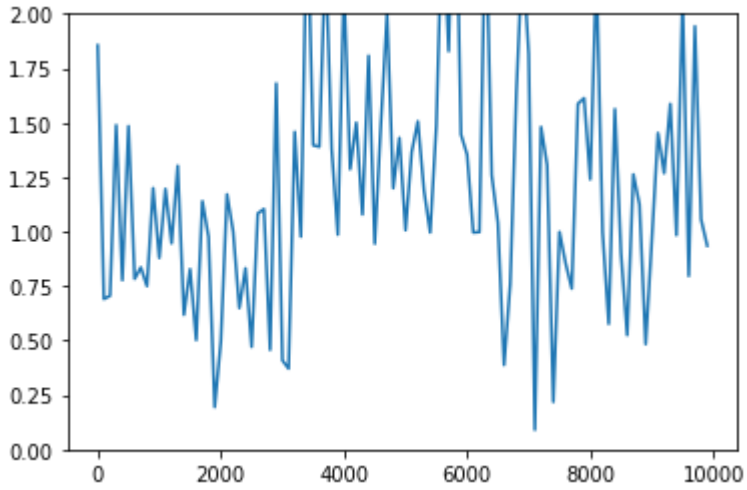
```
------------
iters:5100
Loss:1.365553116383495
Pred:[1 1 1 1 1 1 1 1]
True:[1 1 0 0 1 0 1 1]
114 + 89 = 255
------------
iters:5200
Loss:1.506297622862872
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 0 1 1 1 0]
57 + 117 = 0
------------
iters:5300
Loss:1.1885404879467623
Pred:[0 1 0 0 0 0 0 0]
True:[1 0 0 0 1 0 0 0]
82 + 54 = 64
------------
iters:5400
Loss:0.9958670793581659
Pred:[1 1 0 0 0 0 0 0]
True:[1 0 0 1 1 0 0 0]
89 + 63 = 192
------------
iters:5500
Loss:1.4863978089666126
Pred:[1 1 1 1 1 1 1 1]
True:[1 0 1 1 0 0 1 1]
106 + 73 = 255
------------
iters:5600
Loss:2.494524469537979
Pred:[1 1 1 1 1 1 1 1]
True:[1 0 0 0 0 0 1 1]
125 + 6 = 255
------------
iters:5700
Loss:1.8278560893658167
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 0 0 1 1 0 1]
44 + 33 = 255
------------
iters:5800
Loss:2.8914045236413757
Pred:[0 0 1 1 1 1 1 1]
True:[0 1 0 0 0 0 0 0]
11 + 53 = 63
------------
iters:5900
Loss:1.4464996181772822
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 1 0 0 0 0]
82 + 30 = 0
------------
iters:6000
Loss:1.353390176289948
Pred:[0 0 0 0 0 0 0 1]
True:[0 1 0 0 1 0 1 1]
24 + 51 = 1
------------
```

```
iters:6100
Loss:0.9959297229695215
Pred:[1 1 0 1 0 1 1 1]
True:[1 1 0 1 1 0 1 1]
96 + 123 = 215
------------
iters:6200
Loss:0.9981255986613096
Pred:[0 0 0 0 0 0 0 1]
True:[0 1 0 0 0 1 0 1]
55 + 14 = 1
------------
iters:6300
Loss:2.4999868540962975
Pred:[1 1 1 1 1 1 1 1]
True:[1 1 0 0 0 0 1 0]
67 + 127 = 255
------------
iters:6400
Loss:1.2594316545641755
Pred:[0 1 1 0 1 0 1 0]
True:[1 0 0 0 1 0 1 0]
106 + 32 = 106
------------
iters:6500
Loss:1.0405777442586994
Pred:[0 1 0 0 0 0 0 0]
True:[0 0 1 0 1 0 0 0]
17 + 23 = 64
------------
iters:6600
Loss:0.3876158275737498
Pred:[1 1 1 1 1 1 0 1]
True:[1 1 1 0 1 1 0 1]
125 + 112 = 253
------------
iters:6700
Loss:0.7589680488881082
Pred:[1 1 1 1 1 1 1 0]
True:[1 1 1 0 1 0 1 0]
116 + 118 = 254
------------
iters:6800
Loss:1.6387397362418914
Pred:[0 1 0 0 0 1 0 1]
True:[1 0 0 0 1 0 0 1]
86 + 51 = 69
------------
iters:6900
Loss:2.2863566973126193
Pred:[1 1 0 1 0 1 1 0]
True:[1 0 1 1 1 0 0 0]
78 + 106 = 214
------------
iters:7000
Loss:1.8158084301682262
Pred:[0 0 0 0 0 0 1 1]
True:[0 1 0 1 1 1 1 1]
87 + 8 = 3
------------
iters:7100
```

```
Loss:0.09011013998294944
Pred:[1 0 1 0 0 0 1 1]
True:[1 0 1 0 0 0 1 1]
82 + 81 = 163
------------
iters:7200
Loss:1.4805124778013041
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 0 0 1 1 0]
34 + 36 = 0
------------
iters:7300
Loss:1.3051343363936048
Pred:[0 0 1 1 1 1 1 1]
True:[0 0 1 1 0 0 1 1]
38 + 13 = 63
------------
iters:7400
Loss:0.21807534095536782
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 1 0 0 0 0 0]
32 + 0 = 0
------------
iters:7500
Loss:0.9987680814498255
Pred:[1 1 1 1 1 1 1 1]
True:[1 0 0 1 1 1 1 1]
101 + 58 = 255
------------
iters:7600
Loss:0.854605455315153
Pred:[1 1 1 1 1 1 0 1]
True:[1 1 0 1 1 0 0 1]
121 + 96 = 253
------------
iters:7700
Loss:0.7384479891476289
Pred:[1 1 1 1 1 1 1 0]
True:[1 1 0 0 1 1 1 0]
83 + 123 = 254
------------
iters:7800
Loss:1.5867918037816149
Pred:[0 1 1 1 1 1 1 1]
True:[1 0 0 1 0 1 1 1]
122 + 29 = 127
------------
iters:7900
Loss:1.6123869879929955
Pred:[0 1 1 1 0 1 1 0]
True:[1 0 0 0 1 1 0 0]
22 + 118 = 118
------------
iters:8000
Loss:1.2396651849678908
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 1 1 0 0 0]
44 + 44 = 0
------------
iters:8100
Loss:2.2024469866325824
```

```
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 1 1 1 0 0]
122 + 66 = 0
------------
iters:8200
Loss:0.9986291599375003
Pred:[1 1 1 1 1 0 1 1]
True:[0 1 1 1 1 0 0 1]
32 + 89 = 251
------------
iters:8300
Loss:0.5763278681686153
Pred:[1 1 1 1 0 1 1 0]
True:[0 1 1 1 0 0 1 0]
0 + 114 = 246
------------
iters:8400
Loss:1.5626277170198528
Pred:[0 1 0 0 1 0 0 0]
True:[0 0 1 1 0 0 0 0]
10 + 38 = 72
------------
iters:8500
Loss:0.8997046981035799
Pred:[1 1 1 1 1 1 1 0]
True:[1 1 0 0 1 0 1 0]
83 + 119 = 254
------------
iters:8600
Loss:0.5245576149370273
Pred:[1 1 1 0 1 1 1 0]
True:[1 0 1 0 1 1 1 0]
76 + 98 = 238
------------
iters:8700
Loss:1.262982878325835
Pred:[0 1 1 0 1 1 0 0]
True:[0 1 0 1 1 1 1 0]
78 + 16 = 108
------------
iters:8800
Loss:1.123920795470539
Pred:[1 0 1 1 0 0 0 0]
True:[1 0 0 0 1 0 1 1]
63 + 76 = 176
------------
iters:8900
Loss:0.48332191625734405
Pred:[0 1 0 1 0 1 0 0]
True:[0 1 0 1 0 1 1 0]
43 + 43 = 84
------------
iters:9000
Loss:0.9596909829848015
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 1 1 1 0 1 1]
78 + 45 = 255
------------
iters:9100
Loss:1.452739759005325
Pred:[1 0 1 0 0 0 0 0]
```

```
True:[0 1 1 1 0 1 1 0]
24 + 94 = 160
------------
iters:9200
Loss:1.2682048597267657
Pred:[0 0 0 0 0 0 0 1]
True:[0 1 1 0 1 0 1 1]
70 + 37 = 1
------------
iters:9300
Loss:1.5865042644353355
Pred:[1 1 1 1 0 1 0 1]
True:[0 1 1 0 0 0 0 1]
1 + 96 = 245
------------
iters:9400
Loss:0.9834966633267894
Pred:[1 1 1 1 1 1 1 0]
True:[0 1 0 1 1 1 1 0]
7 + 87 = 254
------------
iters:9500
Loss:2.074475721774039
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 1 1 1 1 0]
108 + 82 = 0
------------
iters:9600
Loss:0.796321068195236
Pred:[1 1 0 0 1 1 0 1]
True:[0 1 0 0 0 1 0 1]
3 + 66 = 205
------------
iters:9700
Loss:1.9426979506440363
Pred:[0 1 1 1 1 1 1 0]
True:[0 0 1 1 0 0 0 0]
23 + 25 = 126
------------
iters:9800
Loss:1.0564090109450355
Pred:[0 1 1 0 0 1 0 0]
True:[0 1 0 1 1 1 0 0]
67 + 25 = 100
------------
iters:9900
Loss:0.9355639213406668
Pred:[1 1 0 0 0 0 1 1]
True:[0 1 1 0 0 1 1 1]
41 + 62 = 195
------------
```



[try] hidden_layer_sizeを変更してみよう

In [6]:

```
# データを用意
# 2進数の桁数
binary_dim = 8
# 最大値 + 1
largest_number = pow(2, binary_dim)
# largest_numberまで2進数を用意
binary = np.unpackbits(np.array([range(largest_number)],dtype=np.uint8).T,axis=1)

input_layer_size = 2
hidden_layer_size = 32
output_layer_size = 1

weight_init_std = 1
learning_rate = 0.1

iters_num = 10000
plot_interval = 100

# ウェイト初期化 （バイアスは簡単のため省略）
W_in = weight_init_std * np.random.randn(input_layer_size, hidden_layer_size)
W_out = weight_init_std * np.random.randn(hidden_layer_size, output_layer_size)
W = weight_init_std * np.random.randn(hidden_layer_size, hidden_layer_size)

# Xavier


# He



# 勾配
W_in_grad = np.zeros_like(W_in)
W_out_grad = np.zeros_like(W_out)
W_grad = np.zeros_like(W)

u = np.zeros((hidden_layer_size, binary_dim + 1))
z = np.zeros((hidden_layer_size, binary_dim + 1))
y = np.zeros((output_layer_size, binary_dim))

delta_out = np.zeros((output_layer_size, binary_dim))
delta = np.zeros((hidden_layer_size, binary_dim + 1))

all_losses = []

for i in range(iters_num):

    # A, B初期化 (a + b = d)
    a_int = np.random.randint(largest_number/2)
    a_bin = binary[a_int] # binary encoding
    b_int = np.random.randint(largest_number/2)
    b_bin = binary[b_int] # binary encoding

    # 正解データ
    d_int = a_int + b_int
    d_bin = binary[d_int]

    # 出力バイナリ
    out_bin = np.zeros_like(d_bin)
```

```
    # 時系列全体の誤差
    all_loss = 0

    # 時系列ループ
    for t in range(binary_dim):
        # 入力値
        X = np.array([a_bin[ - t - 1], b_bin[ - t - 1]]).reshape(1, -1)
        # 時刻tにおける正解データ
        dd = np.array([d_bin[binary_dim - t - 1]])

        u[:,t+1] = np.dot(X, W_in) + np.dot(z[:,t].reshape(1, -1), W)
        z[:,t+1] = functions.sigmoid(u[:,t+1])

        y[:,t] = functions.sigmoid(np.dot(z[:,t+1].reshape(1, -1), W_out))


        #誤差
        loss = functions.mean_squared_error(dd, y[:,t])

        delta_out[:,t] = functions.d_mean_squared_error(dd, y[:,t]) * functions.d_sigmoid(y[:,t
])


        all_loss += loss

        out_bin[binary_dim - t - 1] = np.round(y[:,t])


    for t in range(binary_dim)[::-1]:
        X = np.array([a_bin[-t-1],b_bin[-t-1]]).reshape(1, -1)

        delta[:,t] = (np.dot(delta[:,t+1].T, W.T) + np.dot(delta_out[:,t].T, W_out.T)) * functio
ns.d_sigmoid(u[:,t+1])

        # 勾配更新
        W_out_grad += np.dot(z[:,t+1].reshape(-1,1), delta_out[:,t].reshape(-1,1))
        W_grad += np.dot(z[:,t].reshape(-1,1), delta[:,t].reshape(1,-1))
        W_in_grad += np.dot(X.T, delta[:,t].reshape(1,-1))

    # 勾配適用
    W_in -= learning_rate * W_in_grad
    W_out -= learning_rate * W_out_grad
    W -= learning_rate * W_grad

    W_in_grad *= 0
    W_out_grad *= 0
    W_grad *= 0


    if(i % plot_interval == 0):
        all_losses.append(all_loss)
        print("iters:" + str(i))
        print("Loss:" + str(all_loss))
        print("Pred:" + str(out_bin))
        print("True:" + str(d_bin))
        out_int = 0
        for index,x in enumerate(reversed(out_bin)):
            out_int += x * pow(2, index)
        print(str(a_int) + " + " + str(b_int) + " = " + str(out_int))
        print("------------")

lists = range(0, iters_num, plot_interval)
```

```
plt.plot(lists, all_losses, label="loss")
plt.ylim(0,2.0)
plt.show()
```

```
iters:0
Loss:1.7099911842318891
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 1 0 0 1 1 0]
33 + 69 = 255
------------
iters:100
Loss:0.975529372737529
Pred:[1 1 1 1 0 1 1 1]
True:[0 1 1 0 0 0 1 1]
83 + 16 = 247
------------
iters:200
Loss:1.1644487297156947
Pred:[0 1 1 1 0 0 1 1]
True:[1 0 0 0 1 0 0 1]
23 + 114 = 115
------------
iters:300
Loss:1.5660046635622142
Pred:[1 1 1 1 1 1 1 0]
True:[0 1 1 0 0 0 0 0]
75 + 21 = 254
------------
iters:400
Loss:0.7697875498112977
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 0 0 0 1 0 0]
125 + 71 = 0
------------
iters:500
Loss:0.9781803484104791
Pred:[1 1 1 0 0 0 1 1]
True:[1 1 0 1 0 0 1 1]
98 + 113 = 227
------------
iters:600
Loss:0.9543347424296084
Pred:[0 1 0 0 0 0 0 0]
True:[0 1 0 0 1 1 0 1]
53 + 24 = 64
------------
iters:700
Loss:1.3075864346373667
Pred:[0 0 0 1 1 1 1 1]
True:[0 1 1 0 0 0 0 1]
5 + 92 = 31
------------
iters:800
Loss:1.0587569703077868
Pred:[1 0 0 0 0 0 0 0]
True:[1 1 0 0 0 1 1 1]
92 + 107 = 128
------------
iters:900
Loss:0.9150390830074642
Pred:[0 0 0 1 0 1 1 1]
True:[1 0 0 1 1 1 1 1]
71 + 88 = 23
------------
iters:1000
```

```
Loss:0.818678827741222
Pred:[0 0 1 1 0 0 0 0]
True:[0 0 1 1 1 1 0 1]
19 + 42 = 48
------------
iters:1100
Loss:0.9668626284885959
Pred:[1 1 1 1 0 1 0 0]
True:[0 1 1 1 1 1 0 0]
121 + 3 = 244
------------
iters:1200
Loss:0.8219319907632345
Pred:[1 0 0 0 0 0 0 0]
True:[1 1 0 0 0 0 1 0]
108 + 86 = 128
------------
iters:1300
Loss:1.0339789658664535
Pred:[1 1 0 0 1 1 1 0]
True:[1 0 0 1 0 1 1 0]
51 + 99 = 206
------------
iters:1400
Loss:0.9078437204697866
Pred:[1 0 0 1 0 0 0 1]
True:[1 1 0 0 1 1 0 1]
107 + 98 = 145
------------
iters:1500
Loss:0.9513126969556305
Pred:[0 0 0 1 1 0 1 0]
True:[0 1 0 1 1 1 0 0]
3 + 89 = 26
------------
iters:1600
Loss:1.0867717910605657
Pred:[1 1 1 0 0 1 1 1]
True:[1 0 1 0 1 0 0 1]
101 + 68 = 231
------------
iters:1700
Loss:0.8035918992576898
Pred:[0 1 0 1 1 1 1 1]
True:[0 1 1 0 1 1 1 1]
67 + 44 = 95
------------
iters:1800
Loss:0.5285838001861755
Pred:[0 0 0 0 0 1 0 0]
True:[0 0 0 1 0 1 1 0]
5 + 17 = 4
------------
iters:1900
Loss:0.7632732536725261
Pred:[0 1 1 0 0 0 1 0]
True:[0 1 1 1 0 1 1 0]
84 + 34 = 98
------------
iters:2000
Loss:0.6573491278912984
```

```
Pred:[0 0 1 0 1 1 1 0]
True:[0 0 1 0 0 0 1 0]
17 + 17 = 46
------------
iters:2100
Loss:1.0473620667724408
Pred:[0 0 0 0 0 0 0 1]
True:[0 1 0 1 0 1 1 1]
7 + 80 = 1
------------
iters:2200
Loss:1.2965579902316373
Pred:[1 1 1 0 1 1 1 1]
True:[1 0 0 1 0 0 0 1]
118 + 27 = 239
------------
iters:2300
Loss:0.8508850504608461
Pred:[1 1 0 0 0 0 1 1]
True:[1 0 0 1 0 0 0 1]
34 + 111 = 195
------------
iters:2400
Loss:0.6308751774395102
Pred:[1 0 0 1 1 0 1 1]
True:[1 0 0 1 1 0 1 1]
72 + 83 = 155
------------
iters:2500
Loss:0.9081990569192017
Pred:[0 0 1 0 0 0 0 0]
True:[0 1 1 1 0 0 1 0]
110 + 4 = 32
------------
iters:2600
Loss:0.7321422459849312
Pred:[1 1 1 0 0 0 0 0]
True:[1 0 1 1 0 0 0 0]
60 + 116 = 224
------------
iters:2700
Loss:0.8090746129880255
Pred:[0 1 1 1 1 1 1 0]
True:[0 1 1 0 0 0 1 0]
36 + 62 = 126
------------
iters:2800
Loss:0.7911789215546995
Pred:[1 0 0 1 0 0 0 0]
True:[1 1 0 1 1 0 0 0]
118 + 98 = 144
------------
iters:2900
Loss:1.2661148843679784
Pred:[1 0 1 1 0 1 1 0]
True:[1 1 0 0 0 1 0 0]
105 + 91 = 182
------------
iters:3000
Loss:0.30502880255098674
Pred:[0 1 1 1 1 1 1 0]
```

```
True:[0 1 1 1 1 1 1 0]
116 + 10 = 126
------------
iters:3100
Loss:0.6229421342062681
Pred:[0 0 0 1 0 0 0 1]
True:[0 1 0 1 0 0 0 1]
3 + 78 = 17
------------
iters:3200
Loss:0.8392331752772466
Pred:[0 1 1 1 0 1 0 1]
True:[0 1 1 0 1 0 0 1]
58 + 47 = 117
------------
iters:3300
Loss:0.3048044377084362
Pred:[0 0 0 1 0 0 0 1]
True:[0 0 0 1 1 0 0 1]
13 + 12 = 17
------------
iters:3400
Loss:0.6073866654537986
Pred:[0 1 1 1 1 0 1 1]
True:[0 1 1 1 1 0 1 1]
43 + 80 = 123
------------
iters:3500
Loss:0.33427352820935163
Pred:[0 0 0 0 1 0 0 1]
True:[0 0 0 1 1 0 0 1]
16 + 9 = 9
------------
iters:3600
Loss:0.33072522659192494
Pred:[0 1 0 1 0 1 1 0]
True:[0 1 0 1 0 1 1 0]
59 + 27 = 86
------------
iters:3700
Loss:0.4184869641581752
Pred:[0 1 1 1 0 0 1 1]
True:[0 1 1 1 0 0 1 1]
97 + 18 = 115
------------
iters:3800
Loss:0.07288185568858631
Pred:[0 1 0 1 0 0 0 1]
True:[0 1 0 1 0 0 0 1]
0 + 81 = 81
------------
iters:3900
Loss:0.3292571721870669
Pred:[0 1 0 0 0 1 0 1]
True:[0 1 0 0 1 1 0 1]
3 + 74 = 69
------------
iters:4000
Loss:0.06800427338124874
Pred:[0 1 1 0 1 0 1 0]
True:[0 1 1 0 1 0 1 0]
```

```
65 + 41 = 106
------------
iters:4100
Loss:0.2613110575211153
Pred:[1 0 0 0 1 0 0 1]
True:[1 0 0 0 1 0 0 1]
15 + 122 = 137
------------
iters:4200
Loss:0.20197469945246962
Pred:[1 0 0 0 0 1 0 1]
True:[1 0 0 0 0 1 0 1]
127 + 6 = 133
------------
iters:4300
Loss:0.9432245288223307
Pred:[1 0 1 0 0 0 0 0]
True:[1 1 0 0 0 0 0 0]
83 + 109 = 160
------------
iters:4400
Loss:0.0953659568078607
Pred:[1 1 1 0 1 1 0 0]
True:[1 1 1 0 1 1 0 0]
126 + 110 = 236
------------
iters:4500
Loss:0.017019741641270607
Pred:[0 0 1 0 1 0 1 1]
True:[0 0 1 0 1 0 1 1]
33 + 10 = 43
------------
iters:4600
Loss:0.12369316671669621
Pred:[1 0 0 1 1 1 0 0]
True:[1 0 0 1 1 1 0 0]
75 + 81 = 156
------------
iters:4700
Loss:0.01740991387161899
Pred:[0 0 0 1 1 1 1 0]
True:[0 0 0 1 1 1 1 0]
30 + 0 = 30
------------
iters:4800
Loss:0.05406749910957151
Pred:[0 1 1 1 0 1 0 1]
True:[0 1 1 1 0 1 0 1]
70 + 47 = 117
------------
iters:4900
Loss:0.05512761893958643
Pred:[1 1 1 0 1 1 1 0]
True:[1 1 1 0 1 1 1 0]
111 + 127 = 238
------------
iters:5000
Loss:0.00287318968458683
Pred:[0 0 1 0 0 0 1 0]
True:[0 0 1 0 0 0 1 0]
33 + 1 = 34
```

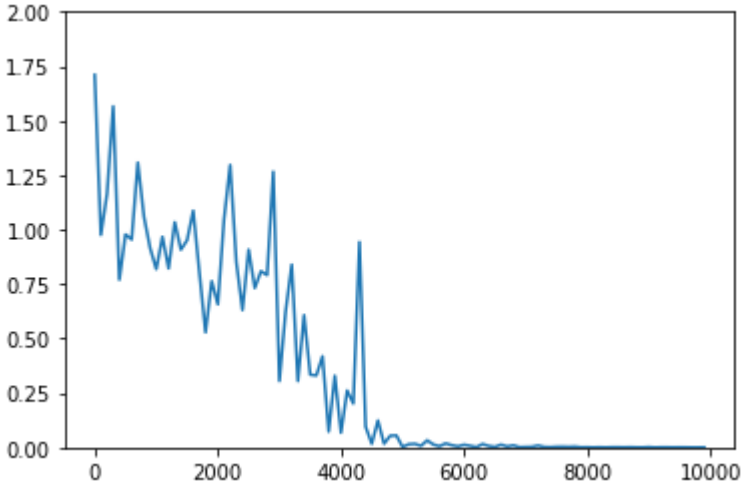
```
------------
iters:5100
Loss:0.014896976519027385
Pred:[1 0 0 0 0 1 1 0]
True:[1 0 0 0 0 1 1 0]
85 + 49 = 134
------------
iters:5200
Loss:0.016731907569333224
Pred:[0 1 0 1 1 0 0 1]
True:[0 1 0 1 1 0 0 1]
44 + 45 = 89
------------
iters:5300
Loss:0.007568534352886077
Pred:[0 1 1 0 1 1 0 0]
True:[0 1 1 0 1 1 0 0]
92 + 16 = 108
------------
iters:5400
Loss:0.032496004211139566
Pred:[1 0 0 1 0 0 1 0]
True:[1 0 0 1 0 0 1 0]
119 + 27 = 146
------------
iters:5500
Loss:0.013709828707088775
Pred:[0 1 0 1 0 1 0 0]
True:[0 1 0 1 0 1 0 0]
45 + 39 = 84
------------
iters:5600
Loss:0.005878401908288687
Pred:[0 1 1 0 1 0 0 0]
True:[0 1 1 0 1 0 0 0]
7 + 97 = 104
------------
iters:5700
Loss:0.01857274308826733
Pred:[1 1 0 0 1 1 1 0]
True:[1 1 0 0 1 1 1 0]
91 + 115 = 206
------------
iters:5800
Loss:0.010474288130514132
Pred:[0 1 0 0 1 1 0 1]
True:[0 1 0 0 1 1 0 1]
6 + 71 = 77
------------
iters:5900
Loss:0.005150642091433643
Pred:[1 0 1 1 0 1 0 1]
True:[1 0 1 1 0 1 0 1]
57 + 124 = 181
------------
iters:6000
Loss:0.012625883466605418
Pred:[1 0 1 1 0 0 1 0]
True:[1 0 1 1 0 0 1 0]
111 + 67 = 178
------------
```

```
iters:6100
Loss:0.0073685723879070315
Pred:[0 1 1 0 1 1 0 1]
True:[0 1 1 0 1 1 0 1]
75 + 34 = 109
------------
iters:6200
Loss:0.0016742661494020914
Pred:[0 1 0 0 1 0 1 0]
True:[0 1 0 0 1 0 1 0]
21 + 53 = 74
------------
iters:6300
Loss:0.015732005219747575
Pred:[1 0 0 0 0 0 1 1]
True:[1 0 0 0 0 0 1 1]
94 + 37 = 131
------------
iters:6400
Loss:0.006892642214088077
Pred:[0 0 1 0 0 0 0 1]
True:[0 0 1 0 0 0 0 1]
18 + 15 = 33
------------
iters:6500
Loss:0.0025738401282609
Pred:[0 1 0 0 0 1 0 0]
True:[0 1 0 0 0 1 0 0]
35 + 33 = 68
------------
iters:6600
Loss:0.012911962137518666
Pred:[0 1 1 1 0 1 1 1]
True:[0 1 1 1 0 1 1 1]
70 + 49 = 119
------------
iters:6700
Loss:0.004614507102527812
Pred:[1 0 0 1 1 0 0 0]
True:[1 0 0 1 1 0 0 0]
76 + 76 = 152
------------
iters:6800
Loss:0.009756618929111692
Pred:[1 1 0 0 0 0 0 0]
True:[1 1 0 0 0 0 0 0]
103 + 89 = 192
------------
iters:6900
Loss:0.0013912733313562822
Pred:[0 1 0 1 1 1 1 1]
True:[0 1 0 1 1 1 1 1]
65 + 30 = 95
------------
iters:7000
Loss:0.002233462718341314
Pred:[0 1 1 1 0 1 0 0]
True:[0 1 1 1 0 1 0 0]
87 + 29 = 116
------------
iters:7100
```

```
Loss:0.002852778154877489
Pred:[0 1 1 0 1 1 1 1]
True:[0 1 1 0 1 1 1 1]
72 + 39 = 111
------------
iters:7200
Loss:0.008830764564298946
Pred:[0 0 1 0 1 1 1 1]
True:[0 0 1 0 1 1 1 1]
22 + 25 = 47
------------
iters:7300
Loss:0.0023641344654816983
Pred:[1 0 0 0 0 1 1 0]
True:[1 0 0 0 0 1 1 0]
30 + 104 = 134
------------
iters:7400
Loss:0.0015429584834086916
Pred:[1 0 1 0 0 1 0 1]
True:[1 0 1 0 0 1 0 1]
69 + 96 = 165
------------
iters:7500
Loss:0.0031387563776547663
Pred:[1 0 0 0 1 1 1 0]
True:[1 0 0 0 1 1 1 0]
15 + 127 = 142
------------
iters:7600
Loss:0.003595749258778409
Pred:[1 0 1 1 0 1 0 1]
True:[1 0 1 1 0 1 0 1]
102 + 79 = 181
------------
iters:7700
Loss:0.0027138587569875837
Pred:[0 1 1 1 0 0 0 0]
True:[0 1 1 1 0 0 0 0]
59 + 53 = 112
------------
iters:7800
Loss:0.003929958485581488
Pred:[0 1 0 0 1 1 1 1]
True:[0 1 0 0 1 1 1 1]
2 + 77 = 79
------------
iters:7900
Loss:0.0008906745279128154
Pred:[1 0 1 0 1 1 1 1]
True:[1 0 1 0 1 1 1 1]
57 + 118 = 175
------------
iters:8000
Loss:0.0012395913307836934
Pred:[1 0 1 0 0 1 0 1]
True:[1 0 1 0 0 1 0 1]
73 + 92 = 165
------------
iters:8100
Loss:0.00045248481887227875
```

```
Pred:[1 0 0 0 0 1 1 0]
True:[1 0 0 0 0 1 1 0]
53 + 81 = 134
------------
iters:8200
Loss:0.001179885382735488
Pred:[0 0 1 0 0 1 1 1]
True:[0 0 1 0 0 1 1 1]
8 + 31 = 39
------------
iters:8300
Loss:0.000463171942783925
Pred:[1 0 1 1 0 1 1 0]
True:[1 0 1 1 0 1 1 0]
105 + 77 = 182
------------
iters:8400
Loss:0.0012837846212889704
Pred:[1 0 1 1 1 0 1 0]
True:[1 0 1 1 1 0 1 0]
111 + 75 = 186
------------
iters:8500
Loss:0.0013624826125921978
Pred:[0 1 0 1 1 1 1 0]
True:[0 1 0 1 1 1 1 0]
12 + 82 = 94
------------
iters:8600
Loss:0.0008447442474532331
Pred:[0 1 1 1 1 1 1 0]
True:[0 1 1 1 1 1 1 0]
47 + 79 = 126
------------
iters:8700
Loss:0.0014104221221657679
Pred:[0 1 1 1 1 1 0 0]
True:[0 1 1 1 1 1 0 0]
124 + 0 = 124
------------
iters:8800
Loss:0.0006872840819004771
Pred:[0 0 1 1 1 1 0 1]
True:[0 0 1 1 1 1 0 1]
3 + 58 = 61
------------
iters:8900
Loss:0.0005696176200872436
Pred:[1 0 0 1 1 1 0 1]
True:[1 0 0 1 1 1 0 1]
105 + 52 = 157
------------
iters:9000
Loss:0.001929592346113461
Pred:[1 1 0 0 0 0 0 1]
True:[1 1 0 0 0 0 0 1]
116 + 77 = 193
------------
iters:9100
Loss:0.0002308478491811303
Pred:[1 0 0 0 0 1 1 0]
```

```
True:[1 0 0 0 0 1 1 0]
113 + 21 = 134
------------
iters:9200
Loss:0.001081036328302397
Pred:[1 0 1 0 1 1 1 1]
True:[1 0 1 0 1 1 1 1]
88 + 87 = 175
------------
iters:9300
Loss:0.0010020212338273987
Pred:[0 1 1 0 1 1 1 0]
True:[0 1 1 0 1 1 1 0]
98 + 12 = 110
------------
iters:9400
Loss:0.0006258760314569332
Pred:[1 0 0 1 0 0 1 0]
True:[1 0 0 1 0 0 1 0]
67 + 79 = 146
------------
iters:9500
Loss:0.0009437982574638379
Pred:[1 0 0 0 1 1 1 0]
True:[1 0 0 0 1 1 1 0]
60 + 82 = 142
------------
iters:9600
Loss:0.0006928540975173552
Pred:[1 0 0 0 0 0 0 1]
True:[1 0 0 0 0 0 0 1]
69 + 60 = 129
------------
iters:9700
Loss:0.00032401805139384906
Pred:[0 1 1 1 0 1 1 0]
True:[0 1 1 1 0 1 1 0]
109 + 9 = 118
------------
iters:9800
Loss:0.00034339334336602186
Pred:[0 1 0 0 1 1 0 1]
True:[0 1 0 0 1 1 0 1]
7 + 70 = 77
------------
iters:9900
Loss:0.0005985374491701768
Pred:[0 1 1 1 1 0 1 0]
True:[0 1 1 1 1 0 1 0]
83 + 39 = 122
------------
```



[try] 重みの初期化方法を変更してみよう

Xavier

In [7]:

```
# データを用意
# 2進数の桁数
binary_dim = 8
# 最大値 + 1
largest_number = pow(2, binary_dim)
# largest_numberまで2進数を用意
binary = np.unpackbits(np.array([range(largest_number)],dtype=np.uint8).T,axis=1)

input_layer_size = 2
hidden_layer_size = 16
output_layer_size = 1

weight_init_std = 1
learning_rate = 0.1

iters_num = 10000
plot_interval = 100

# ウェイト初期化 (バイアスは簡単のため省略)
#W_in = weight_init_std * np.random.randn(input_layer_size, hidden_layer_size)
#W_out = weight_init_std * np.random.randn(hidden_layer_size, output_layer_size)
#W = weight_init_std * np.random.randn(hidden_layer_size, hidden_layer_size)

# Xavier
W_in = weight_init_std * np.random.randn(input_layer_size, hidden_layer_size) * np.sqrt(1 / inpu
t_layer_size)
W_out = weight_init_std * np.random.randn(hidden_layer_size, output_layer_size) * np.sqrt(1 / hi
dden_layer_size)
W = weight_init_std * np.random.randn(hidden_layer_size, hidden_layer_size) * np.sqrt(1 / hidden
_layer_size)

# He



# 勾配
W_in_grad = np.zeros_like(W_in)
W_out_grad = np.zeros_like(W_out)
W_grad = np.zeros_like(W)

u = np.zeros((hidden_layer_size, binary_dim + 1))
z = np.zeros((hidden_layer_size, binary_dim + 1))
y = np.zeros((output_layer_size, binary_dim))

delta_out = np.zeros((output_layer_size, binary_dim))
delta = np.zeros((hidden_layer_size, binary_dim + 1))

all_losses = []

for i in range(iters_num):

    # A, B初期化 (a + b = d)
    a_int = np.random.randint(largest_number/2)
    a_bin = binary[a_int] # binary encoding
    b_int = np.random.randint(largest_number/2)
    b_bin = binary[b_int] # binary encoding

    # 正解データ
    d_int = a_int + b_int
```

```
    d_bin = binary[d_int]

    # 出力バイナリ
    out_bin = np.zeros_like(d_bin)

    # 時系列全体の誤差
    all_loss = 0

    # 時系列ループ
    for t in range(binary_dim):
        # 入力値
        X = np.array([a_bin[ - t - 1], b_bin[ - t - 1]]).reshape(1, -1)
        # 時刻tにおける正解データ
        dd = np.array([d_bin[binary_dim - t - 1]])

        u[:,t+1] = np.dot(X, W_in) + np.dot(z[:,t].reshape(1, -1), W)
        z[:,t+1] = functions.sigmoid(u[:,t+1])

        y[:,t] = functions.sigmoid(np.dot(z[:,t+1].reshape(1, -1), W_out))


        #誤差
        loss = functions.mean_squared_error(dd, y[:,t])

        delta_out[:,t] = functions.d_mean_squared_error(dd, y[:,t]) * functions.d_sigmoid(y[:,t
])

        all_loss += loss

        out_bin[binary_dim - t - 1] = np.round(y[:,t])


    for t in range(binary_dim)[::-1]:
        X = np.array([a_bin[-t-1],b_bin[-t-1]]).reshape(1, -1)

        delta[:,t] = (np.dot(delta[:,t+1].T, W.T) + np.dot(delta_out[:,t].T, W_out.T)) * functio
ns.d_sigmoid(u[:,t+1])

        # 勾配更新
        W_out_grad += np.dot(z[:,t+1].reshape(-1,1), delta_out[:,t].reshape(-1,1))
        W_grad += np.dot(z[:,t].reshape(-1,1), delta[:,t].reshape(1,-1))
        W_in_grad += np.dot(X.T, delta[:,t].reshape(1,-1))

    # 勾配適用
    W_in -= learning_rate * W_in_grad
    W_out -= learning_rate * W_out_grad
    W -= learning_rate * W_grad

    W_in_grad *= 0
    W_out_grad *= 0
    W_grad *= 0


    if(i % plot_interval == 0):
        all_losses.append(all_loss)
        print("iters:" + str(i))
        print("Loss:" + str(all_loss))
        print("Pred:" + str(out_bin))
        print("True:" + str(d_bin))
        out_int = 0
        for index,x in enumerate(reversed(out_bin)):
```

```
            out_int += x * pow(2, index)
        print(str(a_int) + " + " + str(b_int) + " = " + str(out_int))
        print("------------")

lists = range(0, iters_num, plot_interval)
plt.plot(lists, all_losses, label="loss")
plt.ylim(0,2.0)
plt.show()
```

```
iters:0
Loss:1.060664542655887
Pred:[1 1 1 1 1 1 1 1]
True:[1 1 1 0 0 0 1 1]
115 + 112 = 255
------------
iters:100
Loss:0.877274158051635
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 1 1 1 1 0 1]
15 + 110 = 255
------------
iters:200
Loss:1.2104027590038755
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 1 0 0 0 0 0]
73 + 23 = 255
------------
iters:300
Loss:0.9461483607805279
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 1 1 0 0 0]
70 + 18 = 0
------------
iters:400
Loss:1.1574620993684752
Pred:[1 1 1 1 1 1 1 1]
True:[0 0 1 1 0 0 1 0]
3 + 47 = 255
------------
iters:500
Loss:0.9911361844237411
Pred:[1 0 0 1 0 0 0 0]
True:[1 0 1 1 1 0 0 0]
111 + 73 = 144
------------
iters:600
Loss:1.007512517343793
Pred:[1 1 1 1 1 1 1 1]
True:[1 0 0 1 0 1 0 1]
53 + 96 = 255
------------
iters:700
Loss:1.0726333888721
Pred:[1 1 1 1 1 1 1 1]
True:[1 0 0 1 0 0 0 0]
124 + 20 = 255
------------
iters:800
Loss:0.9691155589801264
Pred:[0 1 0 0 0 1 0 0]
True:[1 1 0 1 0 1 0 0]
110 + 102 = 68
------------
iters:900
Loss:0.9254443479383332
Pred:[0 1 1 1 1 1 1 1]
True:[0 1 1 0 1 1 1 1]
50 + 61 = 127
------------
iters:1000
```

```
Loss:1.2193579967236592
Pred:[1 1 1 1 1 1 1 1]
True:[1 0 0 0 0 0 0 1]
54 + 75 = 255
------------
iters:1100
Loss:0.9631019501032817
Pred:[0 0 1 1 1 1 1 1]
True:[0 0 0 1 1 1 1 1]
28 + 3 = 63
------------
iters:1200
Loss:1.0603762022990657
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 1 1 1 1 0]
30 + 64 = 0
------------
iters:1300
Loss:1.0142440625369011
Pred:[0 1 0 0 1 1 0 0]
True:[1 1 0 1 0 0 1 1]
101 + 110 = 76
------------
iters:1400
Loss:1.0238973435968028
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 1 0 0 1]
27 + 110 = 0
------------
iters:1500
Loss:0.8589249326290613
Pred:[0 0 0 1 1 1 1 1]
True:[0 0 0 1 1 0 1 1]
4 + 23 = 31
------------
iters:1600
Loss:0.9396271417069383
Pred:[0 0 1 1 0 0 1 1]
True:[0 1 0 0 0 0 1 1]
9 + 58 = 51
------------
iters:1700
Loss:0.866453314678313
Pred:[1 1 0 0 0 0 0 1]
True:[1 1 1 0 1 0 0 1]
116 + 117 = 193
------------
iters:1800
Loss:1.0443323550446098
Pred:[1 1 1 1 1 0 1 1]
True:[0 1 1 0 1 0 0 1]
13 + 92 = 251
------------
iters:1900
Loss:0.9240319719625796
Pred:[1 0 1 0 0 1 0 0]
True:[1 0 1 1 0 1 0 0]
82 + 98 = 164
------------
iters:2000
Loss:1.0315317263105963
```

```
Pred:[0 1 0 1 1 1 0 0]
True:[1 0 0 1 1 1 1 0]
34 + 124 = 92
------------
iters:2100
Loss:0.850748535018063
Pred:[0 0 0 0 1 1 1 1]
True:[0 1 0 0 1 1 1 1]
73 + 6 = 15
------------
iters:2200
Loss:0.8404710057768374
Pred:[0 1 1 1 1 1 1 1]
True:[0 1 0 1 1 1 1 0]
50 + 44 = 127
------------
iters:2300
Loss:0.7297896751940045
Pred:[0 0 0 0 1 1 1 0]
True:[0 0 0 0 1 0 1 0]
4 + 6 = 14
------------
iters:2400
Loss:0.9813360298009854
Pred:[0 1 0 0 0 1 0 0]
True:[1 0 0 0 1 1 0 0]
34 + 106 = 68
------------
iters:2500
Loss:1.1547782478170687
Pred:[0 1 1 1 1 1 0 1]
True:[1 0 0 0 0 0 0 1]
110 + 19 = 125
------------
iters:2600
Loss:1.0312619589634155
Pred:[1 1 1 1 1 1 1 0]
True:[1 0 0 1 0 1 1 0]
60 + 90 = 254
------------
iters:2700
Loss:0.8619415035248628
Pred:[1 0 0 0 1 0 1 0]
True:[1 0 1 0 1 0 1 1]
70 + 101 = 138
------------
iters:2800
Loss:1.0056780212980108
Pred:[1 0 1 0 1 0 1 1]
True:[1 1 0 1 0 0 1 1]
126 + 85 = 171
------------
iters:2900
Loss:0.5242559814306352
Pred:[0 1 0 0 0 0 1 0]
True:[0 1 0 0 0 0 1 0]
33 + 33 = 66
------------
iters:3000
Loss:0.4110533438107947
Pred:[0 0 0 0 0 0 1 1]
```

```
True:[0 0 0 0 0 0 1 1]
1 + 2 = 3
------------
iters:3100
Loss:0.9056825819264408
Pred:[0 1 1 0 1 0 0 1]
True:[0 1 1 1 0 0 0 1]
20 + 93 = 105
------------
iters:3200
Loss:1.0723650277167804
Pred:[0 1 1 1 0 0 0 0]
True:[1 0 0 0 1 0 0 0]
12 + 124 = 112
------------
iters:3300
Loss:0.7179902210497473
Pred:[0 0 0 1 0 1 1 1]
True:[0 1 0 1 0 1 1 1]
85 + 2 = 23
------------
iters:3400
Loss:0.7097836516863364
Pred:[1 0 1 1 1 0 1 0]
True:[1 0 1 1 0 0 1 0]
108 + 70 = 186
------------
iters:3500
Loss:0.5374184400839184
Pred:[1 1 1 0 1 1 0 0]
True:[1 1 1 0 1 1 0 0]
114 + 122 = 236
------------
iters:3600
Loss:0.7496437168733749
Pred:[0 1 1 0 1 1 0 0]
True:[0 1 1 0 1 1 0 1]
101 + 8 = 108
------------
iters:3700
Loss:0.6442211994990699
Pred:[0 1 1 0 1 0 1 1]
True:[0 1 0 0 1 0 1 1]
56 + 19 = 107
------------
iters:3800
Loss:0.599830644176011
Pred:[0 0 0 0 1 1 0 0]
True:[1 0 0 1 1 1 0 0]
84 + 72 = 12
------------
iters:3900
Loss:0.681016844166206
Pred:[0 1 1 1 1 1 1 1]
True:[0 1 1 1 0 1 1 1]
6 + 113 = 127
------------
iters:4000
Loss:0.6382696340627201
Pred:[1 0 1 1 0 1 1 1]
True:[1 0 1 0 0 0 1 1]
```

```
75 + 88 = 183
------------
iters:4100
Loss:0.40502204526687485
Pred:[0 1 0 1 1 1 1 0]
True:[0 1 0 1 1 1 0 0]
33 + 59 = 94
------------
iters:4200
Loss:1.1255394884503394
Pred:[1 1 1 1 1 1 0 1]
True:[1 0 0 0 0 0 0 1]
99 + 30 = 253
------------
iters:4300
Loss:0.5026893063859631
Pred:[0 1 1 0 1 0 0 0]
True:[0 1 1 1 0 0 0 0]
28 + 84 = 104
------------
iters:4400
Loss:0.3539350815808049
Pred:[0 0 0 1 0 1 1 1]
True:[0 0 0 1 0 0 1 1]
9 + 10 = 23
------------
iters:4500
Loss:0.4479854571553187
Pred:[1 1 0 0 1 1 1 0]
True:[1 0 0 0 1 1 1 0]
16 + 126 = 206
------------
iters:4600
Loss:0.5785167987411105
Pred:[1 0 0 1 0 0 0 0]
True:[1 0 0 1 1 0 0 0]
57 + 95 = 144
------------
iters:4700
Loss:0.1468781952785305
Pred:[0 1 0 1 0 1 1 0]
True:[0 1 0 1 0 1 1 0]
8 + 78 = 86
------------
iters:4800
Loss:0.10394779581458442
Pred:[0 1 1 1 1 1 1 0]
True:[0 1 1 1 1 1 1 0]
2 + 124 = 126
------------
iters:4900
Loss:0.13747676960317345
Pred:[0 1 0 0 1 1 1 0]
True:[0 1 0 0 1 1 1 0]
75 + 3 = 78
------------
iters:5000
Loss:0.5027280064898192
Pred:[0 1 1 1 1 0 1 1]
True:[0 1 1 1 1 0 1 1]
113 + 10 = 123
```

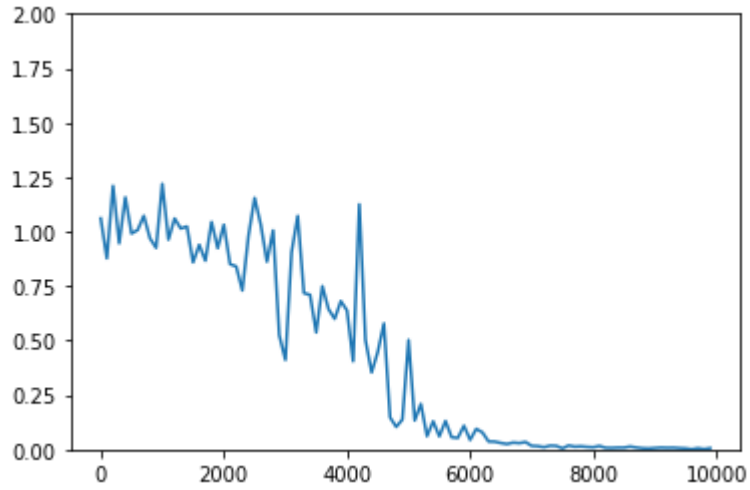
```
------------
iters:5100
Loss:0.1333332895757331
Pred:[0 1 0 0 1 1 0 1]
True:[0 1 0 0 1 1 0 1]
6 + 71 = 77
------------
iters:5200
Loss:0.20898767178738029
Pred:[0 0 1 0 0 1 1 1]
True:[0 0 1 0 0 1 1 1]
18 + 21 = 39
------------
iters:5300
Loss:0.06145521693960945
Pred:[0 1 1 0 0 1 1 1]
True:[0 1 1 0 0 1 1 1]
103 + 0 = 103
------------
iters:5400
Loss:0.12906991111347227
Pred:[1 0 1 0 1 1 0 0]
True:[1 0 1 0 1 1 0 0]
69 + 103 = 172
------------
iters:5500
Loss:0.060530457532689694
Pred:[0 1 1 1 0 0 1 0]
True:[0 1 1 1 0 0 1 0]
58 + 56 = 114
------------
iters:5600
Loss:0.13062160526594138
Pred:[1 0 0 1 1 0 1 1]
True:[1 0 0 1 1 0 1 1]
80 + 75 = 155
------------
iters:5700
Loss:0.05614170157625131
Pred:[1 1 0 1 0 1 1 0]
True:[1 1 0 1 0 1 1 0]
102 + 112 = 214
------------
iters:5800
Loss:0.05301306805609296
Pred:[1 0 1 0 1 1 0 1]
True:[1 0 1 0 1 1 0 1]
94 + 79 = 173
------------
iters:5900
Loss:0.10893911395355989
Pred:[1 0 1 1 0 0 1 0]
True:[1 0 1 1 0 0 1 0]
59 + 119 = 178
------------
iters:6000
Loss:0.04395460648254105
Pred:[0 1 0 1 0 1 1 0]
True:[0 1 0 1 0 1 1 0]
8 + 78 = 86
------------
```

```
iters:6100
Loss:0.09452783531028625
Pred:[1 0 0 1 0 0 1 1]
True:[1 0 0 1 0 0 1 1]
26 + 121 = 147
------------
iters:6200
Loss:0.0783496906144285
Pred:[1 0 0 1 0 1 1 1]
True:[1 0 0 1 0 1 1 1]
78 + 73 = 151
------------
iters:6300
Loss:0.036536571253673523
Pred:[0 1 1 0 0 0 0 1]
True:[0 1 1 0 0 0 0 1]
75 + 22 = 97
------------
iters:6400
Loss:0.03587760783966364
Pred:[1 0 1 0 1 1 1 0]
True:[1 0 1 0 1 1 1 0]
88 + 86 = 174
------------
iters:6500
Loss:0.02978962474369867
Pred:[1 0 0 0 1 0 1 1]
True:[1 0 0 0 1 0 1 1]
65 + 74 = 139
------------
iters:6600
Loss:0.025101186161364957
Pred:[1 1 0 1 0 1 0 1]
True:[1 1 0 1 0 1 0 1]
106 + 107 = 213
------------
iters:6700
Loss:0.03177703042077843
Pred:[0 1 0 0 0 1 0 1]
True:[0 1 0 0 0 1 0 1]
56 + 13 = 69
------------
iters:6800
Loss:0.029256162425125343
Pred:[0 1 0 0 1 0 1 1]
True:[0 1 0 0 1 0 1 1]
32 + 43 = 75
------------
iters:6900
Loss:0.03427783361861606
Pred:[0 1 0 1 1 0 0 0]
True:[0 1 0 1 1 0 0 0]
22 + 66 = 88
------------
iters:7000
Loss:0.01664617977787195
Pred:[0 1 0 1 0 1 0 0]
True:[0 1 0 1 0 1 0 0]
57 + 27 = 84
------------
iters:7100
```

```
Loss:0.015371212247201506
Pred:[0 1 1 1 1 0 0 1]
True:[0 1 1 1 1 0 0 1]
107 + 14 = 121
------------
iters:7200
Loss:0.010670988033006267
Pred:[0 0 1 1 1 1 0 1]
True:[0 0 1 1 1 1 0 1]
17 + 44 = 61
------------
iters:7300
Loss:0.017519185239072744
Pred:[0 1 1 1 0 1 1 0]
True:[0 1 1 1 0 1 1 0]
98 + 20 = 118
------------
iters:7400
Loss:0.01643431095302158
Pred:[0 1 1 1 1 1 1 0]
True:[0 1 1 1 1 1 1 0]
24 + 102 = 126
------------
iters:7500
Loss:0.005336925946469459
Pred:[0 1 1 0 1 0 0 0]
True:[0 1 1 0 1 0 0 0]
103 + 1 = 104
------------
iters:7600
Loss:0.018210944552129817
Pred:[1 0 0 1 1 1 1 0]
True:[1 0 0 1 1 1 1 0]
74 + 84 = 158
------------
iters:7700
Loss:0.013003788261553621
Pred:[0 1 1 1 0 1 1 1]
True:[0 1 1 1 0 1 1 1]
112 + 7 = 119
------------
iters:7800
Loss:0.014545340662973147
Pred:[0 0 1 1 0 0 1 0]
True:[0 0 1 1 0 0 1 0]
46 + 4 = 50
------------
iters:7900
Loss:0.011835109207321514
Pred:[0 1 1 1 1 1 1 0]
True:[0 1 1 1 1 1 1 0]
84 + 42 = 126
------------
iters:8000
Loss:0.010284007854109002
Pred:[0 1 0 0 1 1 0 0]
True:[0 1 0 0 1 1 0 0]
76 + 0 = 76
------------
iters:8100
Loss:0.01613278765188015
```

```
Pred:[0 1 0 1 1 1 0 0]
True:[0 1 0 1 1 1 0 0]
46 + 46 = 92
------------
iters:8200
Loss:0.00777777211508731
Pred:[0 1 0 1 1 1 1 0]
True:[0 1 0 1 1 1 1 0]
76 + 18 = 94
------------
iters:8300
Loss:0.007141091622982163
Pred:[1 0 0 0 1 1 1 1]
True:[1 0 0 0 1 1 1 1]
109 + 34 = 143
------------
iters:8400
Loss:0.008521659004844776
Pred:[0 0 1 0 1 0 1 1]
True:[0 0 1 0 1 0 1 1]
13 + 30 = 43
------------
iters:8500
Loss:0.008527946794010326
Pred:[0 1 1 0 0 1 0 1]
True:[0 1 1 0 0 1 0 1]
60 + 41 = 101
------------
iters:8600
Loss:0.013532815539876078
Pred:[1 0 0 1 0 0 1 0]
True:[1 0 0 1 0 0 1 0]
22 + 124 = 146
------------
iters:8700
Loss:0.008835811609692457
Pred:[0 1 0 1 0 0 1 0]
True:[0 1 0 1 0 0 1 0]
40 + 42 = 82
------------
iters:8800
Loss:0.005557534590150769
Pred:[0 1 1 0 0 1 1 1]
True:[0 1 1 0 0 1 1 1]
89 + 14 = 103
------------
iters:8900
Loss:0.004391563666021166
Pred:[0 1 1 1 0 1 1 1]
True:[0 1 1 1 0 1 1 1]
31 + 88 = 119
------------
iters:9000
Loss:0.006466681901766209
Pred:[0 1 1 0 0 0 1 1]
True:[0 1 1 0 0 0 1 1]
24 + 75 = 99
------------
iters:9100
Loss:0.008397753394384394
Pred:[1 1 0 1 1 1 1 0]
```

```
True:[1 1 0 1 1 1 1 0]
110 + 112 = 222
------------
iters:9200
Loss:0.007462771766095006
Pred:[1 1 0 1 0 0 1 1]
True:[1 1 0 1 0 0 1 1]
88 + 123 = 211
------------
iters:9300
Loss:0.00792357864369963
Pred:[0 1 1 1 0 0 1 1]
True:[0 1 1 1 0 0 1 1]
24 + 91 = 115
------------
iters:9400
Loss:0.006055191978317476
Pred:[0 1 1 1 1 1 0 0]
True:[0 1 1 1 1 1 0 0]
69 + 55 = 124
------------
iters:9500
Loss:0.004760784456676352
Pred:[0 1 1 1 0 0 1 1]
True:[0 1 1 1 0 0 1 1]
69 + 46 = 115
------------
iters:9600
Loss:0.0010441108499656929
Pred:[1 1 1 0 0 0 1 0]
True:[1 1 1 0 0 0 1 0]
113 + 113 = 226
------------
iters:9700
Loss:0.0056781897615943025
Pred:[1 0 1 1 1 1 1 1]
True:[1 0 1 1 1 1 1 1]
64 + 127 = 191
------------
iters:9800
Loss:0.0017058396941361826
Pred:[0 0 1 1 0 0 0 0]
True:[0 0 1 1 0 0 0 0]
11 + 37 = 48
------------
iters:9900
Loss:0.005964222663407769
Pred:[1 0 1 0 1 1 1 0]
True:[1 0 1 0 1 1 1 0]
94 + 80 = 174
------------
```



[try] 重みの初期化方法を変更してみよう

He

In [8]:

```
# データを用意
# 2進数の桁数
binary_dim = 8
# 最大値 + 1
largest_number = pow(2, binary_dim)
# largest_numberまで2進数を用意
binary = np.unpackbits(np.array([range(largest_number)],dtype=np.uint8).T,axis=1)

input_layer_size = 2
hidden_layer_size = 16
output_layer_size = 1

weight_init_std = 1
learning_rate = 0.1

iters_num = 10000
plot_interval = 100

# ウェイト初期化 (バイアスは簡単のため省略)
#W_in = weight_init_std * np.random.randn(input_layer_size, hidden_layer_size)
#W_out = weight_init_std * np.random.randn(hidden_layer_size, output_layer_size)
#W = weight_init_std * np.random.randn(hidden_layer_size, hidden_layer_size)

# Xavier


# He
W_in = weight_init_std * np.random.randn(input_layer_size, hidden_layer_size) * np.sqrt(2 / inpu
t_layer_size)
W_out = weight_init_std * np.random.randn(hidden_layer_size, output_layer_size) * np.sqrt(2 / hi
dden_layer_size)
W = weight_init_std * np.random.randn(hidden_layer_size, hidden_layer_size) * np.sqrt(2 / hidden
_layer_size)


# 勾配
W_in_grad = np.zeros_like(W_in)
W_out_grad = np.zeros_like(W_out)
W_grad = np.zeros_like(W)

u = np.zeros((hidden_layer_size, binary_dim + 1))
z = np.zeros((hidden_layer_size, binary_dim + 1))
y = np.zeros((output_layer_size, binary_dim))

delta_out = np.zeros((output_layer_size, binary_dim))
delta = np.zeros((hidden_layer_size, binary_dim + 1))

all_losses = []

for i in range(iters_num):

    # A, B初期化 (a + b = d)
    a_int = np.random.randint(largest_number/2)
    a_bin = binary[a_int] # binary encoding
    b_int = np.random.randint(largest_number/2)
    b_bin = binary[b_int] # binary encoding

    # 正解データ
    d_int = a_int + b_int
```

```
    d_bin = binary[d_int]

    # 出力バイナリ
    out_bin = np.zeros_like(d_bin)

    # 時系列全体の誤差
    all_loss = 0

    # 時系列ループ
    for t in range(binary_dim):
        # 入力値
        X = np.array([a_bin[ - t - 1], b_bin[ - t - 1]]).reshape(1, -1)
        # 時刻tにおける正解データ
        dd = np.array([d_bin[binary_dim - t - 1]])

        u[:,t+1] = np.dot(X, W_in) + np.dot(z[:,t].reshape(1, -1), W)
        z[:,t+1] = functions.sigmoid(u[:,t+1])

        y[:,t] = functions.sigmoid(np.dot(z[:,t+1].reshape(1, -1), W_out))


        #誤差
        loss = functions.mean_squared_error(dd, y[:,t])

        delta_out[:,t] = functions.d_mean_squared_error(dd, y[:,t]) * functions.d_sigmoid(y[:,t
])

        all_loss += loss

        out_bin[binary_dim - t - 1] = np.round(y[:,t])


    for t in range(binary_dim)[::-1]:
        X = np.array([a_bin[-t-1],b_bin[-t-1]]).reshape(1, -1)

        delta[:,t] = (np.dot(delta[:,t+1].T, W.T) + np.dot(delta_out[:,t].T, W_out.T)) * functio
ns.d_sigmoid(u[:,t+1])

        # 勾配更新
        W_out_grad += np.dot(z[:,t+1].reshape(-1,1), delta_out[:,t].reshape(-1,1))
        W_grad += np.dot(z[:,t].reshape(-1,1), delta[:,t].reshape(1,-1))
        W_in_grad += np.dot(X.T, delta[:,t].reshape(1,-1))

    # 勾配適用
    W_in -= learning_rate * W_in_grad
    W_out -= learning_rate * W_out_grad
    W -= learning_rate * W_grad

    W_in_grad *= 0
    W_out_grad *= 0
    W_grad *= 0


    if(i % plot_interval == 0):
        all_losses.append(all_loss)
        print("iters:" + str(i))
        print("Loss:" + str(all_loss))
        print("Pred:" + str(out_bin))
        print("True:" + str(d_bin))
        out_int = 0
        for index,x in enumerate(reversed(out_bin)):
```

```
            out_int += x * pow(2, index)
        print(str(a_int) + " + " + str(b_int) + " = " + str(out_int))
        print("------------")

lists = range(0, iters_num, plot_interval)
plt.plot(lists, all_losses, label="loss")
plt.ylim(0,2.0)
plt.show()
```

```
iters:0
Loss:1.4412751396148635
Pred:[1 1 1 1 1 1 1 1]
True:[1 1 0 0 0 1 1 0]
82 + 116 = 255
------------
iters:100
Loss:1.009266848512064
Pred:[1 1 1 1 0 1 1 1]
True:[0 0 1 0 1 1 0 1]
27 + 18 = 247
------------
iters:200
Loss:0.9567393199027676
Pred:[1 1 1 1 1 1 1 1]
True:[0 0 1 1 0 1 1 1]
20 + 35 = 255
------------
iters:300
Loss:0.9636932646829031
Pred:[0 1 0 0 0 1 0 0]
True:[0 1 1 0 0 0 0 0]
35 + 61 = 68
------------
iters:400
Loss:1.1646791130586693
Pred:[1 1 1 1 1 1 1 1]
True:[1 0 0 0 1 0 1 0]
59 + 79 = 255
------------
iters:500
Loss:1.010060254899016
Pred:[1 1 0 1 1 1 1 1]
True:[1 0 1 0 1 0 0 1]
110 + 59 = 223
------------
iters:600
Loss:0.9951377262385492
Pred:[0 0 1 1 0 1 1 0]
True:[0 1 0 0 0 0 1 0]
27 + 39 = 54
------------
iters:700
Loss:1.048104466697784
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 1 1 0 1]
80 + 29 = 0
------------
iters:800
Loss:1.0508221562705924
Pred:[1 1 0 0 1 1 1 0]
True:[1 0 0 1 0 0 0 0]
71 + 73 = 206
------------
iters:900
Loss:1.1058430530166314
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 1 0 1 0 1]
64 + 21 = 0
------------
iters:1000
```

```
Loss:1.0254257615692375
Pred:[0 0 1 0 0 0 1 0]
True:[1 0 0 0 1 0 1 1]
17 + 122 = 34
------------
iters:1100
Loss:1.1479832235838625
Pred:[1 1 1 0 1 1 1 0]
True:[1 0 0 1 0 0 1 1]
100 + 47 = 238
------------
iters:1200
Loss:0.884297675377383
Pred:[0 0 0 0 0 1 0 0]
True:[0 0 1 1 0 1 1 0]
2 + 52 = 4
------------
iters:1300
Loss:0.8184410909327638
Pred:[0 1 0 0 0 0 0 0]
True:[0 0 1 1 0 0 0 0]
32 + 16 = 64
------------
iters:1400
Loss:1.0218362910076628
Pred:[1 1 1 1 1 1 1 0]
True:[1 0 1 0 0 1 1 0]
124 + 42 = 254
------------
iters:1500
Loss:0.9320522626691561
Pred:[0 0 1 1 0 1 1 0]
True:[0 0 1 1 1 0 0 1]
27 + 30 = 54
------------
iters:1600
Loss:0.9619287258512584
Pred:[1 1 1 0 1 1 1 1]
True:[1 0 1 1 1 1 0 1]
103 + 86 = 239
------------
iters:1700
Loss:0.9035129464060867
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 1 0 0 0 0 0]
118 + 106 = 0
------------
iters:1800
Loss:0.9442335709635987
Pred:[0 1 0 0 0 0 0 0]
True:[1 0 0 0 1 1 1 0]
102 + 40 = 64
------------
iters:1900
Loss:1.1943446827532553
Pred:[1 1 1 1 1 1 1 1]
True:[0 1 1 0 0 1 0 0]
7 + 93 = 255
------------
iters:2000
Loss:1.1604795686604499
```

```
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 0 0 1 0 1]
57 + 108 = 0
------------
iters:2100
Loss:1.1478453971586025
Pred:[1 1 1 1 0 1 1 0]
True:[1 0 0 0 1 0 0 1]
95 + 42 = 246
------------
iters:2200
Loss:0.7361792913293118
Pred:[0 0 1 0 1 0 1 0]
True:[0 0 1 0 1 1 1 0]
21 + 25 = 42
------------
iters:2300
Loss:1.0321298139696076
Pred:[1 1 1 1 1 1 1 1]
True:[1 1 0 0 0 0 1 1]
111 + 84 = 255
------------
iters:2400
Loss:0.9345631264295288
Pred:[1 0 1 0 0 1 1 1]
True:[1 0 1 0 1 0 0 1]
82 + 87 = 167
------------
iters:2500
Loss:1.0925979000473782
Pred:[1 0 1 1 0 1 0 1]
True:[1 1 0 0 1 0 0 1]
127 + 74 = 181
------------
iters:2600
Loss:0.843584932238035
Pred:[1 1 1 0 1 0 1 0]
True:[1 0 0 0 1 0 1 0]
21 + 117 = 234
------------
iters:2700
Loss:0.8052274383512765
Pred:[0 1 0 0 0 1 1 1]
True:[0 1 1 0 0 1 1 1]
103 + 0 = 71
------------
iters:2800
Loss:0.7792856740694823
Pred:[0 1 0 1 0 1 1 1]
True:[0 1 1 0 0 1 1 1]
43 + 60 = 87
------------
iters:2900
Loss:0.7581255385853484
Pred:[0 0 1 0 0 1 0 0]
True:[1 0 0 1 0 1 0 0]
58 + 90 = 36
------------
iters:3000
Loss:0.8174072507219681
Pred:[0 0 0 1 1 1 1 1]
```

```
True:[0 1 0 0 1 1 0 1]
12 + 65 = 31
------------
iters:3100
Loss:0.8863476798323604
Pred:[1 1 0 1 1 1 0 1]
True:[1 1 0 1 0 0 0 1]
111 + 98 = 221
------------
iters:3200
Loss:1.0401045129469313
Pred:[0 1 1 1 1 1 0 1]
True:[0 1 1 1 0 0 0 1]
3 + 110 = 125
------------
iters:3300
Loss:0.5108176973132221
Pred:[0 0 0 0 0 1 0 0]
True:[0 1 0 1 0 1 0 0]
2 + 82 = 4
------------
iters:3400
Loss:0.8815710399865814
Pred:[1 0 1 1 1 0 0 1]
True:[1 1 0 0 1 0 1 1]
95 + 108 = 185
------------
iters:3500
Loss:1.060094356994484
Pred:[0 1 1 1 1 1 1 0]
True:[0 1 1 1 0 0 1 0]
35 + 79 = 126
------------
iters:3600
Loss:0.46315601526060457
Pred:[1 1 0 1 1 1 0 1]
True:[1 0 0 1 1 1 0 1]
38 + 119 = 221
------------
iters:3700
Loss:0.7047527612485178
Pred:[0 1 1 1 1 1 0 1]
True:[0 1 1 1 0 0 0 1]
42 + 71 = 125
------------
iters:3800
Loss:0.795910212169439
Pred:[0 1 1 1 1 1 0 0]
True:[0 1 0 0 0 0 0 0]
62 + 2 = 124
------------
iters:3900
Loss:0.22959454169034155
Pred:[0 1 1 1 1 0 1 0]
True:[0 1 1 1 1 0 1 0]
12 + 110 = 122
------------
iters:4000
Loss:0.47451972260919106
Pred:[0 1 1 0 0 0 0 0]
True:[0 1 1 0 1 0 0 0]
```

```
90 + 14 = 96
------------
iters:4100
Loss:0.31408726556801597
Pred:[1 0 1 0 1 1 0 0]
True:[1 0 1 0 1 1 0 0]
59 + 113 = 172
------------
iters:4200
Loss:1.1637193069881873
Pred:[1 0 1 0 0 0 0 1]
True:[1 0 0 1 1 1 1 1]
126 + 33 = 161
------------
iters:4300
Loss:0.41356153326039796
Pred:[1 0 0 0 1 1 1 0]
True:[1 0 0 1 1 1 1 0]
46 + 112 = 142
------------
iters:4400
Loss:0.8769656788096626
Pred:[1 1 0 1 1 1 0 0]
True:[1 0 0 1 0 0 0 0]
35 + 109 = 220
------------
iters:4500
Loss:0.27355602462799833
Pred:[0 1 0 1 0 0 1 1]
True:[0 1 0 1 0 0 1 1]
52 + 31 = 83
------------
iters:4600
Loss:0.21936673952845645
Pred:[1 0 1 1 0 1 0 1]
True:[1 0 1 1 0 1 0 1]
56 + 125 = 181
------------
iters:4700
Loss:0.21554409694231402
Pred:[0 1 0 0 1 0 1 1]
True:[0 1 0 0 1 0 1 1]
31 + 44 = 75
------------
iters:4800
Loss:0.156935839505135
Pred:[0 1 1 1 1 0 0 0]
True:[0 1 1 1 1 0 0 0]
87 + 33 = 120
------------
iters:4900
Loss:0.12817591797078917
Pred:[1 0 1 0 0 0 1 1]
True:[1 0 1 0 0 0 1 1]
80 + 83 = 163
------------
iters:5000
Loss:0.0645727391575417
Pred:[0 0 1 1 1 1 0 0]
True:[0 0 1 1 1 1 0 0]
11 + 49 = 60
```

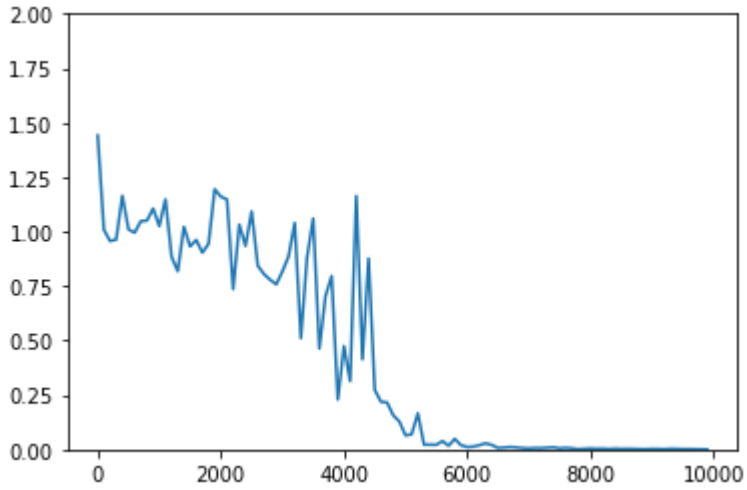
```
------------
iters:5100
Loss:0.07017697577726044
Pred:[1 0 1 1 0 0 0 1]
True:[1 0 1 1 0 0 0 1]
56 + 121 = 177
------------
iters:5200
Loss:0.16582474552461982
Pred:[1 0 0 0 1 1 1 0]
True:[1 0 0 0 1 1 1 0]
39 + 103 = 142
------------
iters:5300
Loss:0.02237142699283128
Pred:[0 0 1 0 1 0 1 0]
True:[0 0 1 0 1 0 1 0]
10 + 32 = 42
------------
iters:5400
Loss:0.022072184803463322
Pred:[0 0 1 1 1 1 1 0]
True:[0 0 1 1 1 1 1 0]
45 + 17 = 62
------------
iters:5500
Loss:0.02157495547374917
Pred:[0 0 1 0 0 1 1 0]
True:[0 0 1 0 0 1 1 0]
36 + 2 = 38
------------
iters:5600
Loss:0.03825101658804902
Pred:[0 1 0 1 1 0 1 1]
True:[0 1 0 1 1 0 1 1]
72 + 19 = 91
------------
iters:5700
Loss:0.017367086726499194
Pred:[1 1 0 0 0 1 0 1]
True:[1 1 0 0 0 1 0 1]
85 + 112 = 197
------------
iters:5800
Loss:0.048705817055552596
Pred:[1 0 0 0 0 0 1 1]
True:[1 0 0 0 0 0 1 1]
93 + 38 = 131
------------
iters:5900
Loss:0.01997730993337107
Pred:[1 0 1 0 0 1 0 0]
True:[1 0 1 0 0 1 0 0]
44 + 120 = 164
------------
iters:6000
Loss:0.011427776370750691
Pred:[0 0 1 0 1 0 1 0]
True:[0 0 1 0 1 0 1 0]
32 + 10 = 42
------------
```

```
iters:6100
Loss:0.013122260206534167
Pred:[0 0 1 1 0 1 0 1]
True:[0 0 1 1 0 1 0 1]
14 + 39 = 53
------------
iters:6200
Loss:0.019865007648554093
Pred:[0 1 0 0 1 0 0 0]
True:[0 1 0 0 1 0 0 0]
53 + 19 = 72
------------
iters:6300
Loss:0.028232072477598687
Pred:[0 1 0 1 0 0 0 0]
True:[0 1 0 1 0 0 0 0]
65 + 15 = 80
------------
iters:6400
Loss:0.0210202702516626
Pred:[0 1 1 1 1 1 0 0]
True:[0 1 1 1 1 1 0 0]
102 + 22 = 124
------------
iters:6500
Loss:0.007165918885207171
Pred:[1 1 1 0 0 1 0 0]
True:[1 1 1 0 0 1 0 0]
107 + 121 = 228
------------
iters:6600
Loss:0.008872036698543027
Pred:[0 1 1 0 1 1 0 1]
True:[0 1 1 0 1 1 0 1]
0 + 109 = 109
------------
iters:6700
Loss:0.011313506823774395
Pred:[1 0 0 0 0 0 0 1]
True:[1 0 0 0 0 0 0 1]
87 + 42 = 129
------------
iters:6800
Loss:0.009216158881845316
Pred:[0 1 0 0 0 1 0 0]
True:[0 1 0 0 0 1 0 0]
11 + 57 = 68
------------
iters:6900
Loss:0.006309031249733778
Pred:[0 1 1 1 1 1 0 1]
True:[0 1 1 1 1 1 0 1]
42 + 83 = 125
------------
iters:7000
Loss:0.004659598489291333
Pred:[1 0 1 1 1 1 0 0]
True:[1 0 1 1 1 1 0 0]
121 + 67 = 188
------------
iters:7100
```

```
Loss:0.006531588483645987
Pred:[1 0 0 1 0 0 1 1]
True:[1 0 0 1 0 0 1 1]
115 + 32 = 147
----------
iters:7200
Loss:0.006287997783401657
Pred:[0 1 0 1 1 0 0 0]
True:[0 1 0 1 1 0 0 0]
4 + 84 = 88
----------
iters:7300
Loss:0.007956603755288394
Pred:[1 0 0 0 1 0 0 0]
True:[1 0 0 0 1 0 0 0]
55 + 81 = 136
----------
iters:7400
Loss:0.009794279264362589
Pred:[0 1 0 1 0 1 1 1]
True:[0 1 0 1 0 1 1 1]
70 + 17 = 87
----------
iters:7500
Loss:0.005415647826536991
Pred:[0 1 0 0 1 0 1 1]
True:[0 1 0 0 1 0 1 1]
35 + 40 = 75
----------
iters:7600
Loss:0.00732141188783723
Pred:[1 0 1 0 0 1 1 1]
True:[1 0 1 0 0 1 1 1]
92 + 75 = 167
----------
iters:7700
Loss:0.005203515833524264
Pred:[1 0 0 0 0 0 1 1]
True:[1 0 0 0 0 0 1 1]
39 + 92 = 131
----------
iters:7800
Loss:0.0015029099423258766
Pred:[1 0 0 1 1 1 1 0]
True:[1 0 0 1 1 1 1 0]
109 + 49 = 158
----------
iters:7900
Loss:0.0030306106247680333
Pred:[0 1 1 1 1 0 0 0]
True:[0 1 1 1 1 0 0 0]
47 + 73 = 120
----------
iters:8000
Loss:0.00467253375382867
Pred:[1 0 0 0 0 0 0 0]
True:[1 0 0 0 0 0 0 0]
44 + 84 = 128
----------
iters:8100
Loss:0.003249061289410127
```

```
Pred:[0 0 1 1 0 0 0 1]
True:[0 0 1 1 0 0 0 1]
38 + 11 = 49
------------
iters:8200
Loss:0.003944659183752557
Pred:[0 0 1 0 0 1 0 1]
True:[0 0 1 0 0 1 0 1]
6 + 31 = 37
------------
iters:8300
Loss:0.002094131983625128
Pred:[0 1 0 1 1 0 0 1]
True:[0 1 0 1 1 0 0 1]
83 + 6 = 89
------------
iters:8400
Loss:0.003999851068299974
Pred:[0 1 1 1 1 0 1 1]
True:[0 1 1 1 1 0 1 1]
26 + 97 = 123
------------
iters:8500
Loss:0.0026177588357899607
Pred:[0 1 0 1 0 0 0 0]
True:[0 1 0 1 0 0 0 0]
55 + 25 = 80
------------
iters:8600
Loss:0.00301720003290917
Pred:[0 1 0 1 0 1 0 1]
True:[0 1 0 1 0 1 0 1]
30 + 55 = 85
------------
iters:8700
Loss:0.002765440714839154
Pred:[0 0 0 0 1 1 1 1]
True:[0 0 0 0 1 1 1 1]
15 + 0 = 15
------------
iters:8800
Loss:0.0017345549170806917
Pred:[1 0 1 0 0 0 0 0]
True:[1 0 1 0 0 0 0 0]
89 + 71 = 160
------------
iters:8900
Loss:0.001310869811853547
Pred:[0 1 1 1 1 0 0 0]
True:[0 1 1 1 1 0 0 0]
17 + 103 = 120
------------
iters:9000
Loss:0.0024919191683100876
Pred:[0 1 1 0 1 1 0 1]
True:[0 1 1 0 1 1 0 1]
10 + 99 = 109
------------
iters:9100
Loss:0.002503684426910758
Pred:[1 0 0 1 0 0 0 1]
```

```
True:[1 0 0 1 0 0 0 1]
112 + 33 = 145
------------
iters:9200
Loss:0.0012942596922245174
Pred:[1 1 0 0 1 0 0 1]
True:[1 1 0 0 1 0 0 1]
105 + 96 = 201
------------
iters:9300
Loss:0.003105560031606557
Pred:[0 0 1 1 0 0 1 0]
True:[0 0 1 1 0 0 1 0]
38 + 12 = 50
------------
iters:9400
Loss:0.002866742780892124
Pred:[1 1 1 1 0 1 1 0]
True:[1 1 1 1 0 1 1 0]
126 + 120 = 246
------------
iters:9500
Loss:0.0018291032578065846
Pred:[1 1 1 1 0 0 1 1]
True:[1 1 1 1 0 0 1 1]
123 + 120 = 243
------------
iters:9600
Loss:0.0018810184959483637
Pred:[0 1 0 1 0 0 0 0]
True:[0 1 0 1 0 0 0 0]
56 + 24 = 80
------------
iters:9700
Loss:0.001438772004288136
Pred:[1 1 1 0 0 0 0 1]
True:[1 1 1 0 0 0 0 1]
111 + 114 = 225
------------
iters:9800
Loss:0.0007881151303112412
Pred:[1 0 1 0 0 1 0 0]
True:[1 0 1 0 0 1 0 0]
43 + 121 = 164
------------
iters:9900
Loss:0.00025632507621349885
Pred:[0 0 1 1 0 0 1 0]
True:[0 0 1 1 0 0 1 0]
49 + 1 = 50
------------
```



[try] 中間層の活性化関数を変更してみよう

ReLU(勾配爆発を確認しよう)

In [3]:

```
import sys, os
sys.path.append(os.pardir)  # 親ディレクトリのファイルをインポートするための設定
import numpy as np
from common import functions
import matplotlib.pyplot as plt

# データを用意
# 2進数の桁数
binary_dim = 8
# 最大値 + 1
largest_number = pow(2, binary_dim)
# largest_numberまで2進数を用意
binary = np.unpackbits(np.array([range(largest_number)],dtype=np.uint8).T,axis=1)

input_layer_size = 2
hidden_layer_size = 16
output_layer_size = 1

weight_init_std = 1
learning_rate = 0.1

iters_num = 10000
plot_interval = 100

# ウェイト初期化 (バイアスは簡単のため省略)
W_in = weight_init_std * np.random.randn(input_layer_size, hidden_layer_size)
W_out = weight_init_std * np.random.randn(hidden_layer_size, output_layer_size)
W = weight_init_std * np.random.randn(hidden_layer_size, hidden_layer_size)

# Xavier


# He


# 勾配
W_in_grad = np.zeros_like(W_in)
W_out_grad = np.zeros_like(W_out)
W_grad = np.zeros_like(W)

u = np.zeros((hidden_layer_size, binary_dim + 1))
z = np.zeros((hidden_layer_size, binary_dim + 1))
y = np.zeros((output_layer_size, binary_dim))

delta_out = np.zeros((output_layer_size, binary_dim))
delta = np.zeros((hidden_layer_size, binary_dim + 1))

all_losses = []

for i in range(iters_num):

    # A, B初期化 (a + b = d)
    a_int = np.random.randint(largest_number/2)
    a_bin = binary[a_int] # binary encoding
    b_int = np.random.randint(largest_number/2)
    b_bin = binary[b_int] # binary encoding

    # 正解データ
    d_int = a_int + b_int
```

```
    d_bin = binary[d_int]

    # 出力バイナリ
    out_bin = np.zeros_like(d_bin)

    # 時系列全体の誤差
    all_loss = 0

    # 時系列ループ
    for t in range(binary_dim):
        # 入力値
        X = np.array([a_bin[ - t - 1], b_bin[ - t - 1]]).reshape(1, -1)
        # 時刻tにおける正解データ
        dd = np.array([d_bin[binary_dim - t - 1]])

        u[:,t+1] = np.dot(X, W_in) + np.dot(z[:,t].reshape(1, -1), W)
        z[:,t+1] = functions.relu(u[:,t+1])

        y[:,t] = functions.sigmoid(np.dot(z[:,t+1].reshape(1, -1), W_out))


        #誤差
        loss = functions.mean_squared_error(dd, y[:,t])

        delta_out[:,t] = functions.d_mean_squared_error(dd, y[:,t]) * functions.d_relu(y[:,t])

        all_loss += loss

        out_bin[binary_dim - t - 1] = np.round(y[:,t])


    for t in range(binary_dim)[::-1]:
        X = np.array([a_bin[-t-1],b_bin[-t-1]]).reshape(1, -1)

        delta[:,t] = (np.dot(delta[:,t+1].T, W.T) + np.dot(delta_out[:,t].T, W_out.T)) * functio
ns.d_sigmoid(u[:,t+1])

        # 勾配更新
        W_out_grad += np.dot(z[:,t+1].reshape(-1,1), delta_out[:,t].reshape(-1,1))
        W_grad += np.dot(z[:,t].reshape(-1,1), delta[:,t].reshape(1,-1))
        W_in_grad += np.dot(X.T, delta[:,t].reshape(1,-1))

    #print(W_grad)
    # 勾配適用
    W_in -= learning_rate * W_in_grad
    W_out -= learning_rate * W_out_grad
    W -= learning_rate * W_grad

    W_in_grad *= 0
    W_out_grad *= 0
    W_grad *= 0


    if(i % plot_interval == 0):
        all_losses.append(all_loss)
        print("iters:" + str(i))
        print("Loss:" + str(all_loss))
        print("Pred:" + str(out_bin))
        print("True:" + str(d_bin))
        out_int = 0
        for index,x in enumerate(reversed(out_bin)):
```

```
            out_int += x * pow(2, index)
        print(str(a_int) + " + " + str(b_int) + " = " + str(out_int))
        print("------------")

lists = range(0, iters_num, plot_interval)
plt.plot(lists, all_losses, label="loss")
plt.ylim(0,2.0)
plt.show()
```

```
iters:0
Loss:1.8937903258840378
Pred:[1 1 1 1 1 0 0 1]
True:[0 1 0 0 1 0 1 1]
5 + 70 = 249
------------
iters:100
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 0 0 0 0 1 1]
109 + 86 = 0
------------
iters:200
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 0 1 1 0]
27 + 107 = 0
------------
iters:300
Loss:3.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 1 1 1 1]
101 + 10 = 0
------------
iters:400
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 1 1 1 0]
113 + 29 = 0
------------
iters:500
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 1 1 0 0 1]
79 + 42 = 0
------------
iters:600
Loss:3.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 1 1 0 1 1]
113 + 74 = 0
------------
iters:700
Loss:1.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 1 0 0 0]
25 + 111 = 0
------------
iters:800
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 0 0 0 0 1 1]
91 + 104 = 0
------------
iters:900
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 0 1 0 1 0]
37 + 37 = 0
------------
iters:1000
```

```
Loss:3.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 1 0 1 1 1]
114 + 5 = 0
------------
iters:1100
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 0 0 1 0 1]
31 + 38 = 0
------------
iters:1200
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 0 1 1 0 1]
50 + 123 = 0
------------
iters:1300
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 0 0 0 1]
52 + 45 = 0
------------
iters:1400
Loss:1.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 1 1 0 0 0 0]
1 + 47 = 0
------------
iters:1500
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 0 0 0 0 1]
117 + 44 = 0
------------
iters:1600
Loss:1.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 0 0 0 0 0 0]
127 + 65 = 0
------------
iters:1700
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 0 0 0 1]
88 + 9 = 0
------------
iters:1800
Loss:1.75
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 1 1 0 0]
112 + 28 = 0
------------
iters:1900
Loss:1.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 0 1 0 0 0]
23 + 49 = 0
------------
iters:2000
Loss:2.5
```

```
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 1 0 1 0 1]
100 + 81 = 0
------------
iters:2100
Loss:1.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 1 0 0 0]
45 + 91 = 0
------------
iters:2200
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 1 0 0 1 1]
97 + 82 = 0
------------
iters:2300
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 0 1 1 0 0]
103 + 69 = 0
------------
iters:2400
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 1 0 1 1]
109 + 30 = 0
------------
iters:2500
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 1 0 1 0 0 1]
37 + 4 = 0
------------
iters:2600
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 1 0 0 0 1]
77 + 4 = 0
------------
iters:2700
Loss:1.25
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 0 0 1 0 0]
64 + 4 = 0
------------
iters:2800
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 1 1 0 1]
102 + 39 = 0
------------
iters:2900
Loss:1.625
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 0 1 0 0]
86 + 14 = 0
------------
iters:3000
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
```

```
True:[1 1 1 0 0 1 0 1]
103 + 126 = 0
------------
iters:3100
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 1 1 0 0]
45 + 63 = 0
------------
iters:3200
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 0 1 1 1]
86 + 65 = 0
------------
iters:3300
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 0 0 1 1 1 0]
123 + 83 = 0
------------
iters:3400
Loss:2.125
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 0 0 1 1 0 0]
126 + 78 = 0
------------
iters:3500
Loss:1.625
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 1 0 1 0]
22 + 116 = 0
------------
iters:3600
Loss:1.125
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 0 0 0 0]
22 + 74 = 0
------------
iters:3700
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 1 1 0 0]
107 + 49 = 0
------------
iters:3800
Loss:3.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 1 1 1 0 1]
87 + 38 = 0
------------
iters:3900
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 0 0 1 0 1 1]
127 + 76 = 0
------------
iters:4000
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 1 0 1 0 1 0]
```

```
37 + 5 = 0
------------
iters:4100
Loss:2.125
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 1 0 1 1 0]
54 + 32 = 0
------------
iters:4200
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 1 0 0 1 0]
23 + 91 = 0
------------
iters:4300
Loss:2.125
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 0 0 1 1 0 0]
118 + 86 = 0
------------
iters:4400
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 1 0 0 1 1 1]
3 + 36 = 0
------------
iters:4500
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 1 0 1 0 0]
103 + 77 = 0
------------
iters:4600
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 0 0 1 1]
118 + 13 = 0
------------
iters:4700
Loss:1.125
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 0 0 1 0]
96 + 34 = 0
------------
iters:4800
Loss:2.625
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 1 1 0 1 0]
76 + 110 = 0
------------
iters:4900
Loss:1.625
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 1 0 0 0 0]
70 + 106 = 0
------------
iters:5000
Loss:1.75
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 0 0 1 0 0]
64 + 100 = 0
```

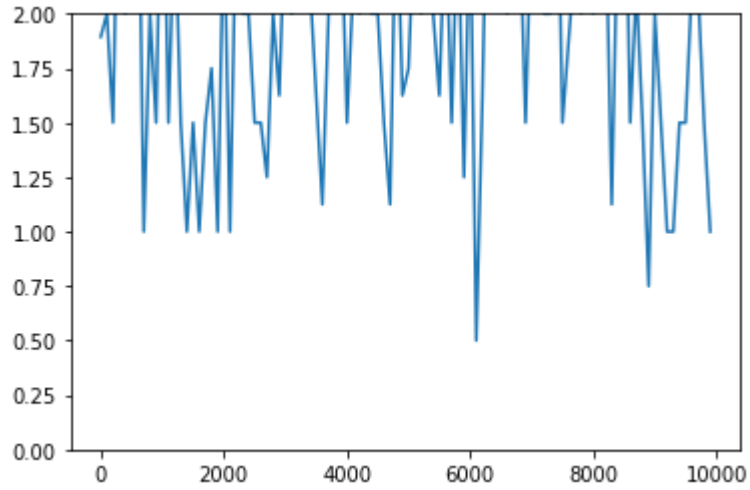
```
------------
iters:5100
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 0 1 1 1 0]
79 + 95 = 0
------------
iters:5200
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 0 1 1 1 0 1]
8 + 21 = 0
------------
iters:5300
Loss:2.125
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 1 1 1 0]
110 + 32 = 0
------------
iters:5400
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 0 1 0 1]
67 + 82 = 0
------------
iters:5500
Loss:1.625
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 0 0 1 0]
46 + 100 = 0
------------
iters:5600
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 1 1 1 1 0]
73 + 21 = 0
------------
iters:5700
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 0 0 1 1 0]
57 + 13 = 0
------------
iters:5800
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 1 0 1 1]
101 + 54 = 0
------------
iters:5900
Loss:1.25
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 0 1 0 0]
112 + 20 = 0
------------
iters:6000
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 1 1 1 0 1 1]
30 + 29 = 0
------------
```

```
iters:6100
Loss:0.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 0 1 0 0 0 0]
15 + 1 = 0
------------
iters:6200
Loss:1.625
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 0 0 1 0]
64 + 34 = 0
------------
iters:6300
Loss:3.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 1 1 1 1 0]
105 + 85 = 0
------------
iters:6400
Loss:3.125
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 1 1 1 1 0]
126 + 64 = 0
------------
iters:6500
Loss:2.625
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 1 0 1 0 1 0]
126 + 108 = 0
------------
iters:6600
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 1 0 0 0 1 0]
121 + 105 = 0
------------
iters:6700
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 1 1 1 1]
52 + 91 = 0
------------
iters:6800
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 1 0 1 1]
116 + 39 = 0
------------
iters:6900
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 1 1 0 0 0 1]
1 + 48 = 0
------------
iters:7000
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 0 0 1 0 1 1]
77 + 126 = 0
------------
iters:7100
```

```
Loss:2.25
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 1 0 1 0 0]
64 + 52 = 0
------------
iters:7200
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 0 0 1 1 0]
75 + 91 = 0
------------
iters:7300
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 0 1 0 0 1]
43 + 126 = 0
------------
iters:7400
Loss:3.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 1 1 1 1]
62 + 97 = 0
------------
iters:7500
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 0 0 1 0 0]
127 + 37 = 0
------------
iters:7600
Loss:1.875
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 1 0 0 0]
64 + 40 = 0
------------
iters:7700
Loss:2.25
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 0 1 1 0 0 0]
108 + 108 = 0
------------
iters:7800
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 0 0 1 1 1]
27 + 44 = 0
------------
iters:7900
Loss:2.125
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 0 1 1 0 0 0]
114 + 102 = 0
------------
iters:8000
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 1 0 0 1]
80 + 25 = 0
------------
iters:8100
Loss:2.5
```

```
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 0 0 1 1 1]
118 + 49 = 0
------------
iters:8200
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 1 0 1 1 1]
50 + 37 = 0
------------
iters:8300
Loss:1.125
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 0 0 1 0 0]
42 + 26 = 0
------------
iters:8400
Loss:2.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 0 1 1 1]
120 + 31 = 0
------------
iters:8500
Loss:3.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 1 1 0 1 1]
109 + 78 = 0
------------
iters:8600
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 0 1 0 1]
84 + 49 = 0
------------
iters:8700
Loss:2.125
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 1 1 0 0 1 0]
78 + 100 = 0
------------
iters:8800
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 1 1 0 0 0 0 0]
109 + 115 = 0
------------
iters:8900
Loss:0.75
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 0 0 0 0]
68 + 60 = 0
------------
iters:9000
Loss:2.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 0 0 1 0 1 1]
54 + 21 = 0
------------
iters:9100
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
```

```
True:[0 0 1 1 0 1 0 0]
35 + 17 = 0
------------
iters:9200
Loss:1.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 0 0 0 0]
63 + 81 = 0
------------
iters:9300
Loss:1.0
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 0 1 0 0 1 0]
17 + 1 = 0
------------
iters:9400
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 0 0 1 0]
105 + 41 = 0
------------
iters:9500
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[0 1 1 0 0 0 1 0]
61 + 37 = 0
------------
iters:9600
Loss:2.125
Pred:[0 0 0 0 0 0 0 0]
True:[0 0 1 0 1 1 1 0]
38 + 8 = 0
------------
iters:9700
Loss:2.125
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 1 0 1 1 0]
96 + 54 = 0
------------
iters:9800
Loss:1.5
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 0 1 1 0]
19 + 115 = 0
------------
iters:9900
Loss:1.0
Pred:[0 0 0 0 0 0 0 0]
True:[1 0 0 0 0 1 0 0]
105 + 27 = 0
------------
```



[try] 中間層の活性化関数を変更してみよう

tanh(numpyにtanhが用意されている。導関数をd_tanhとして作成しよう)

In [10]:

```
def d_tanh(x):
    return 1/(np.cosh(x)**2)


# データを用意
# 2進数の桁数
binary_dim = 8
# 最大値 + 1
largest_number = pow(2, binary_dim)
# largest_numberまで2進数を用意
binary = np.unpackbits(np.array([range(largest_number)],dtype=np.uint8).T,axis=1)

input_layer_size = 2
hidden_layer_size = 16
output_layer_size = 1

weight_init_std = 1
learning_rate = 0.1

iters_num = 10000
plot_interval = 100

# ウェイト初期化 (バイアスは簡単のため省略)
W_in = weight_init_std * np.random.randn(input_layer_size, hidden_layer_size)
W_out = weight_init_std * np.random.randn(hidden_layer_size, output_layer_size)
W = weight_init_std * np.random.randn(hidden_layer_size, hidden_layer_size)

# Xavier


# He



# 勾配
W_in_grad = np.zeros_like(W_in)
W_out_grad = np.zeros_like(W_out)
W_grad = np.zeros_like(W)

u = np.zeros((hidden_layer_size, binary_dim + 1))
z = np.zeros((hidden_layer_size, binary_dim + 1))
y = np.zeros((output_layer_size, binary_dim))

delta_out = np.zeros((output_layer_size, binary_dim))
delta = np.zeros((hidden_layer_size, binary_dim + 1))

all_losses = []

for i in range(iters_num):

    # A, B初期化 (a + b = d)
    a_int = np.random.randint(largest_number/2)
    a_bin = binary[a_int] # binary encoding
    b_int = np.random.randint(largest_number/2)
    b_bin = binary[b_int] # binary encoding

    # 正解データ
    d_int = a_int + b_int
    d_bin = binary[d_int]
```

```
    # 出力バイナリ
    out_bin = np.zeros_like(d_bin)

    # 時系列全体の誤差
    all_loss = 0

    # 時系列ループ
    for t in range(binary_dim):
        # 入力値
        X = np.array([a_bin[ - t - 1], b_bin[ - t - 1]]).reshape(1, -1)
        # 時刻tにおける正解データ
        dd = np.array([d_bin[binary_dim - t - 1]])

        u[:,t+1] = np.dot(X, W_in) + np.dot(z[:,t].reshape(1, -1), W)
        z[:,t+1] = np.tanh(u[:,t+1])

        y[:,t] = functions.sigmoid(np.dot(z[:,t+1].reshape(1, -1), W_out))


        #誤差
        loss = functions.mean_squared_error(dd, y[:,t])

        delta_out[:,t] = functions.d_mean_squared_error(dd, y[:,t]) * functions.d_sigmoid(y[:,t
])


        all_loss += loss

        out_bin[binary_dim - t - 1] = np.round(y[:,t])


    for t in range(binary_dim)[::-1]:
        X = np.array([a_bin[-t-1],b_bin[-t-1]]).reshape(1, -1)

        delta[:,t] = (np.dot(delta[:,t+1].T, W.T) + np.dot(delta_out[:,t].T, W_out.T)) * d_tanh(
u[:,t+1])

        # 勾配更新
        W_out_grad += np.dot(z[:,t+1].reshape(-1,1), delta_out[:,t].reshape(-1,1))
        W_grad += np.dot(z[:,t].reshape(-1,1), delta[:,t].reshape(1,-1))
        W_in_grad += np.dot(X.T, delta[:,t].reshape(1,-1))

    # 勾配適用
    W_in -= learning_rate * W_in_grad
    W_out -= learning_rate * W_out_grad
    W -= learning_rate * W_grad

    W_in_grad *= 0
    W_out_grad *= 0
    W_grad *= 0


    if(i % plot_interval == 0):
        all_losses.append(all_loss)
        print("iters:" + str(i))
        print("Loss:" + str(all_loss))
        print("Pred:" + str(out_bin))
        print("True:" + str(d_bin))
        out_int = 0
        for index,x in enumerate(reversed(out_bin)):
            out_int += x * pow(2, index)
```

```
        print(str(a_int) + " + " + str(b_int) + " = " + str(out_int))
        print("------------")

lists = range(0, iters_num, plot_interval)
plt.plot(lists, all_losses, label="loss")
plt.ylim(0,2.0)
plt.show()
```

```
iters:0
Loss:0.6925362302610845
Pred:[0 1 0 1 1 0 0 0]
True:[0 1 0 1 1 0 1 0]
65 + 25 = 88
------------
iters:100
Loss:0.9458466079453569
Pred:[1 0 1 1 1 0 0 0]
True:[1 0 1 1 0 0 0 1]
93 + 84 = 184
------------
iters:200
Loss:1.1520251707372227
Pred:[1 1 0 0 0 0 1 1]
True:[0 1 1 0 1 1 0 0]
57 + 51 = 195
------------
iters:300
Loss:1.0534818413241667
Pred:[1 1 0 1 1 0 1 1]
True:[0 1 0 0 1 1 0 1]
16 + 61 = 219
------------
iters:400
Loss:0.8852797346755161
Pred:[1 0 0 0 1 1 0 0]
True:[0 1 0 0 1 1 0 0]
12 + 64 = 140
------------
iters:500
Loss:0.9541403600149749
Pred:[1 0 0 1 1 0 0 1]
True:[1 0 1 0 1 0 0 1]
111 + 58 = 153
------------
iters:600
Loss:1.0662377156895257
Pred:[1 1 1 0 0 0 1 0]
True:[1 0 0 1 0 1 1 0]
69 + 81 = 226
------------
iters:700
Loss:1.1874530624530741
Pred:[1 0 0 0 0 0 0 1]
True:[0 1 1 1 1 1 1 1]
9 + 118 = 129
------------
iters:800
Loss:0.7144636290904944
Pred:[0 1 0 1 1 0 0 1]
True:[0 1 1 1 0 0 0 1]
54 + 59 = 89
------------
iters:900
Loss:0.646281179456394
Pred:[0 1 0 0 0 1 1 0]
True:[0 1 0 0 1 1 1 0]
47 + 31 = 70
------------
iters:1000
```

```
Loss:0.9678183887754339
Pred:[0 1 0 1 0 1 1 0]
True:[1 0 1 1 0 0 1 0]
72 + 106 = 86
------------
iters:1100
Loss:1.362306362973361
Pred:[0 1 1 0 0 1 1 1]
True:[1 0 1 0 1 0 0 1]
53 + 116 = 103
------------
iters:1200
Loss:0.7928910933989788
Pred:[0 0 0 1 0 1 1 0]
True:[1 0 1 0 1 1 1 0]
59 + 115 = 22
------------
iters:1300
Loss:1.065991465395114
Pred:[0 1 0 0 1 1 0 1]
True:[1 0 0 1 1 1 1 1]
124 + 35 = 77
------------
iters:1400
Loss:0.8238955245423631
Pred:[0 1 1 0 0 1 1 0]
True:[0 0 1 1 0 0 1 0]
23 + 27 = 102
------------
iters:1500
Loss:0.7238202991430316
Pred:[1 0 1 1 0 1 0 1]
True:[1 0 1 0 0 0 1 1]
88 + 75 = 181
------------
iters:1600
Loss:0.958790625585384
Pred:[0 0 1 1 0 0 1 1]
True:[0 0 1 1 1 1 0 1]
33 + 28 = 51
------------
iters:1700
Loss:1.2781129690525679
Pred:[0 1 1 0 0 1 1 1]
True:[0 0 0 1 1 1 1 1]
11 + 20 = 103
------------
iters:1800
Loss:0.8295927478023275
Pred:[1 0 1 1 0 0 0 1]
True:[1 1 1 0 1 0 0 1]
124 + 109 = 177
------------
iters:1900
Loss:0.8675022495477249
Pred:[1 1 0 0 1 1 1 0]
True:[0 1 1 0 1 0 1 0]
49 + 57 = 206
------------
iters:2000
Loss:0.9686281094180246
```

```
Pred:[0 1 1 0 0 1 1 0]
True:[0 1 1 0 1 0 0 0]
79 + 25 = 102
------------
iters:2100
Loss:0.9938810806655826
Pred:[0 0 1 1 0 0 1 0]
True:[0 1 0 1 1 0 1 0]
27 + 63 = 50
------------
iters:2200
Loss:1.028094946928669
Pred:[1 0 0 0 1 0 0 1]
True:[0 1 0 1 0 0 0 1]
6 + 75 = 137
------------
iters:2300
Loss:0.7770749046674514
Pred:[1 1 1 0 0 1 1 1]
True:[1 0 1 1 0 1 0 1]
86 + 95 = 231
------------
iters:2400
Loss:0.8855338081587907
Pred:[0 1 1 0 0 1 1 0]
True:[1 0 1 0 0 0 1 0]
117 + 45 = 102
------------
iters:2500
Loss:0.6787136416642279
Pred:[1 0 0 0 0 1 1 0]
True:[1 0 0 0 1 0 1 0]
21 + 117 = 134
------------
iters:2600
Loss:0.7884466164811884
Pred:[0 0 1 0 0 0 1 0]
True:[0 1 1 0 1 0 1 0]
2 + 104 = 34
------------
iters:2700
Loss:0.6771997932177587
Pred:[0 1 1 0 0 0 1 1]
True:[0 1 0 0 0 1 1 1]
48 + 23 = 99
------------
iters:2800
Loss:0.6918913444875869
Pred:[1 0 1 1 1 0 1 1]
True:[1 0 1 0 0 1 1 1]
90 + 77 = 187
------------
iters:2900
Loss:1.0376420698998166
Pred:[1 0 1 0 0 1 1 0]
True:[1 0 0 1 0 1 0 0]
71 + 77 = 166
------------
iters:3000
Loss:0.9922709034533456
Pred:[0 1 1 0 1 0 1 0]
```

```
True:[1 0 0 0 1 0 1 0]
108 + 30 = 106
------------
iters:3100
Loss:1.073074800693962
Pred:[0 1 1 0 0 1 0 1]
True:[0 1 0 1 1 0 1 1]
89 + 2 = 101
------------
iters:3200
Loss:1.0185446449143403
Pred:[0 1 1 0 0 1 1 0]
True:[0 0 0 1 1 0 1 0]
21 + 5 = 102
------------
iters:3300
Loss:0.8885845604578338
Pred:[0 0 1 1 0 1 1 0]
True:[0 0 1 0 1 1 0 0]
7 + 37 = 54
------------
iters:3400
Loss:0.6482468670672279
Pred:[0 1 1 0 1 0 0 1]
True:[0 1 0 0 1 0 0 1]
10 + 63 = 105
------------
iters:3500
Loss:0.8898112653560621
Pred:[0 0 1 1 0 0 0 1]
True:[0 0 1 1 1 1 1 1]
61 + 2 = 49
------------
iters:3600
Loss:0.8614134778335648
Pred:[0 0 1 1 0 0 1 1]
True:[1 0 0 0 0 0 1 1]
46 + 85 = 51
------------
iters:3700
Loss:1.0355322677156202
Pred:[0 1 1 0 1 0 0 0]
True:[1 0 0 0 0 0 0 0]
10 + 118 = 104
------------
iters:3800
Loss:0.708718444432084
Pred:[1 1 0 0 1 0 1 0]
True:[0 1 1 0 1 0 1 0]
22 + 84 = 202
------------
iters:3900
Loss:0.8942251104656767
Pred:[1 0 1 0 1 0 1 0]
True:[1 1 0 0 1 0 0 0]
123 + 77 = 170
------------
iters:4000
Loss:0.8218306406455279
Pred:[0 1 1 1 0 1 1 0]
True:[0 1 1 1 1 1 1 0]
```

```
48 + 78 = 118
------------
iters:4100
Loss:0.8939970707444093
Pred:[1 1 1 0 1 1 0 1]
True:[1 0 1 1 0 1 1 1]
119 + 64 = 237
------------
iters:4200
Loss:0.9417332318208068
Pred:[1 0 1 1 0 0 0 1]
True:[1 0 0 1 1 1 1 1]
67 + 92 = 177
------------
iters:4300
Loss:0.9580439127283165
Pred:[0 0 1 0 1 1 0 1]
True:[0 1 0 1 1 0 0 1]
71 + 18 = 45
------------
iters:4400
Loss:0.9037687183136793
Pred:[1 1 0 1 0 1 1 0]
True:[0 0 1 1 1 1 1 0]
52 + 10 = 214
------------
iters:4500
Loss:0.8611972459011543
Pred:[0 1 1 0 1 1 0 1]
True:[0 0 1 1 1 1 0 1]
17 + 44 = 109
------------
iters:4600
Loss:0.8563219972884315
Pred:[0 0 1 1 0 0 1 0]
True:[0 1 1 0 1 0 1 0]
98 + 8 = 50
------------
iters:4700
Loss:0.7216862909570531
Pred:[0 0 1 1 0 0 0 1]
True:[0 0 1 1 1 0 1 1]
49 + 10 = 49
------------
iters:4800
Loss:0.49218282553299575
Pred:[1 0 1 0 0 0 1 1]
True:[1 0 1 0 0 1 1 1]
62 + 105 = 163
------------
iters:4900
Loss:1.2804081819040478
Pred:[0 1 1 0 0 0 1 0]
True:[1 0 0 0 1 0 0 0]
107 + 29 = 98
------------
iters:5000
Loss:0.8329800373863671
Pred:[0 1 1 0 1 1 0 1]
True:[0 1 0 0 0 0 0 1]
64 + 1 = 109
```

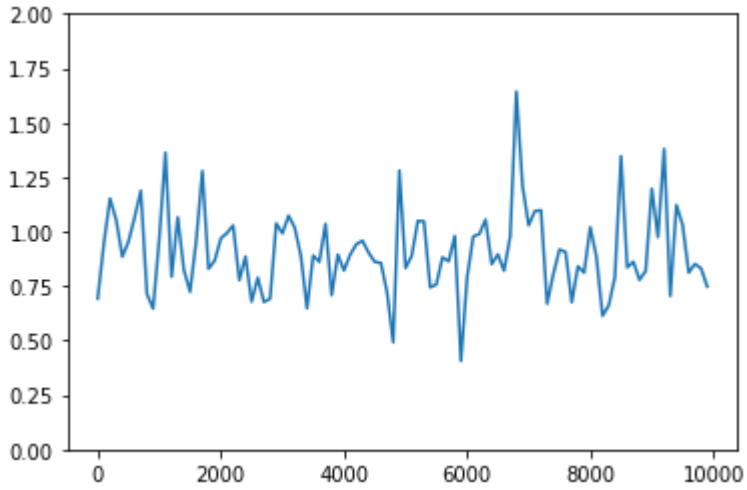
```
------------
iters:5100
Loss:0.8887669344225051
Pred:[0 1 1 0 0 0 1 1]
True:[0 0 0 0 1 0 1 1]
2 + 9 = 99
------------
iters:5200
Loss:1.0488416058113035
Pred:[0 1 1 0 1 1 0 1]
True:[0 1 0 1 1 0 1 1]
1 + 90 = 109
------------
iters:5300
Loss:1.047018253918431
Pred:[0 0 1 0 0 1 0 1]
True:[0 1 0 1 1 0 0 1]
31 + 58 = 37
------------
iters:5400
Loss:0.7441889441786715
Pred:[0 0 1 1 1 1 0 0]
True:[1 1 0 0 1 1 0 0]
106 + 98 = 60
------------
iters:5500
Loss:0.7569877745135357
Pred:[0 1 1 0 0 0 1 1]
True:[0 0 1 0 0 1 0 1]
33 + 4 = 99
------------
iters:5600
Loss:0.8819985697232507
Pred:[0 1 0 1 1 0 0 0]
True:[1 0 0 1 1 0 0 0]
88 + 64 = 88
------------
iters:5700
Loss:0.8636298459456936
Pred:[0 1 0 0 1 1 1 0]
True:[0 0 1 1 1 1 0 0]
7 + 53 = 78
------------
iters:5800
Loss:0.9803275527391078
Pred:[1 0 1 1 0 0 0 1]
True:[1 0 0 0 1 1 1 1]
73 + 70 = 177
------------
iters:5900
Loss:0.4061774314553634
Pred:[0 1 1 0 0 0 0 0]
True:[0 0 1 0 0 0 0 0]
13 + 19 = 96
------------
iters:6000
Loss:0.7893698301654415
Pred:[0 0 1 0 1 1 1 0]
True:[1 0 1 1 0 1 1 0]
80 + 102 = 46
------------
```

```
iters:6100
Loss:0.97790152381324
Pred:[1 1 1 0 1 0 0 0]
True:[0 1 1 0 0 0 0 0]
44 + 52 = 232
------------
iters:6200
Loss:0.989135221159565
Pred:[1 0 1 1 0 1 1 1]
True:[0 0 1 1 1 0 1 1]
43 + 16 = 183
------------
iters:6300
Loss:1.0555298142046579
Pred:[0 1 1 0 0 0 1 0]
True:[1 0 0 0 0 0 1 0]
74 + 56 = 98
------------
iters:6400
Loss:0.8510329013497427
Pred:[1 1 1 0 1 0 0 1]
True:[1 0 0 1 0 0 0 1]
116 + 29 = 233
------------
iters:6500
Loss:0.8959025626522786
Pred:[0 0 0 1 0 0 1 0]
True:[0 0 1 0 0 0 1 0]
16 + 18 = 18
------------
iters:6600
Loss:0.820675660305128
Pred:[0 0 0 0 1 1 0 1]
True:[1 0 0 0 0 1 1 1]
47 + 88 = 13
------------
iters:6700
Loss:0.9776793662225794
Pred:[1 0 0 1 1 1 1 0]
True:[1 0 1 1 0 0 1 0]
87 + 91 = 158
------------
iters:6800
Loss:1.6431020456424446
Pred:[0 0 1 1 1 0 0 0]
True:[1 0 0 0 0 1 1 0]
63 + 71 = 56
------------
iters:6900
Loss:1.2102378253399575
Pred:[0 1 0 0 0 0 1 0]
True:[1 0 0 0 1 1 1 0]
74 + 68 = 66
------------
iters:7000
Loss:1.028926776417571
Pred:[1 0 1 0 1 1 1 1]
True:[0 0 1 1 0 0 1 1]
44 + 7 = 175
------------
iters:7100
```

```
Loss:1.0942436246752272
Pred:[1 1 0 0 0 1 1 0]
True:[1 0 1 0 1 0 1 0]
127 + 43 = 198
------------
iters:7200
Loss:1.09796486840258
Pred:[1 0 0 1 0 0 0 0]
True:[0 1 0 0 0 0 0 0]
52 + 12 = 144
------------
iters:7300
Loss:0.6690051416806805
Pred:[0 0 1 1 0 1 0 0]
True:[0 0 1 0 0 1 0 0]
20 + 16 = 52
------------
iters:7400
Loss:0.8040761519586218
Pred:[0 0 1 1 0 0 0 1]
True:[1 0 1 0 0 1 0 1]
50 + 115 = 49
------------
iters:7500
Loss:0.9179059193922242
Pred:[0 1 1 0 0 1 1 1]
True:[1 0 1 0 0 0 0 1]
44 + 117 = 103
------------
iters:7600
Loss:0.9063571725431417
Pred:[1 0 1 1 0 1 0 1]
True:[1 1 0 0 1 1 1 1]
127 + 80 = 181
------------
iters:7700
Loss:0.6758332655392139
Pred:[1 0 1 0 1 1 1 1]
True:[0 0 1 0 1 1 1 1]
41 + 6 = 175
------------
iters:7800
Loss:0.8406674038148422
Pred:[0 1 1 0 0 1 0 1]
True:[0 1 1 0 0 0 0 1]
7 + 90 = 101
------------
iters:7900
Loss:0.8115445957474512
Pred:[1 1 1 0 1 1 1 0]
True:[0 0 1 0 0 1 1 0]
17 + 21 = 238
------------
iters:8000
Loss:1.0200580184809551
Pred:[1 1 1 0 1 1 0 1]
True:[1 0 0 1 1 0 1 1]
124 + 31 = 237
------------
iters:8100
Loss:0.8856166509431529
```

```
Pred:[1 0 0 1 1 1 0 1]
True:[1 1 1 0 0 0 0 1]
123 + 102 = 157
------------
iters:8200
Loss:0.6138756243474452
Pred:[1 0 0 0 0 0 0 0]
True:[1 1 0 0 0 0 0 0]
99 + 93 = 128
------------
iters:8300
Loss:0.6608987369362741
Pred:[1 0 1 0 0 1 1 1]
True:[1 0 1 0 0 1 1 1]
43 + 124 = 167
------------
iters:8400
Loss:0.7905266716394653
Pred:[1 0 0 0 0 0 0 0]
True:[1 0 1 0 1 1 1 0]
87 + 87 = 128
------------
iters:8500
Loss:1.3465447984540575
Pred:[0 1 1 0 0 1 1 1]
True:[0 0 0 1 1 0 0 1]
16 + 9 = 103
------------
iters:8600
Loss:0.835681229089225
Pred:[1 1 0 0 0 1 1 1]
True:[1 1 0 0 1 0 0 1]
100 + 101 = 199
------------
iters:8700
Loss:0.8620189534077974
Pred:[0 0 0 1 0 0 0 1]
True:[0 1 0 0 1 1 0 1]
73 + 4 = 17
------------
iters:8800
Loss:0.7784843038093073
Pred:[0 1 1 1 0 1 0 1]
True:[0 1 0 1 0 1 0 1]
81 + 4 = 117
------------
iters:8900
Loss:0.8191725687099874
Pred:[1 1 1 0 0 1 1 1]
True:[0 0 1 1 0 0 1 1]
6 + 45 = 231
------------
iters:9000
Loss:1.1966165910747144
Pred:[0 0 0 1 0 0 0 1]
True:[0 1 1 0 1 1 0 1]
31 + 78 = 17
------------
iters:9100
Loss:0.9748714430567795
Pred:[0 0 1 1 0 0 0 1]
```

```
True:[1 0 0 0 1 0 0 1]
12 + 125 = 49
------------
iters:9200
Loss:1.3797486450084502
Pred:[1 0 0 0 1 0 0 0]
True:[0 1 1 1 1 1 1 0]
77 + 49 = 136
------------
iters:9300
Loss:0.7046761192912385
Pred:[1 1 0 1 0 1 0 1]
True:[0 1 0 1 0 1 1 1]
41 + 46 = 213
------------
iters:9400
Loss:1.121694348159171
Pred:[0 0 0 1 0 0 0 1]
True:[0 1 0 0 1 1 1 1]
47 + 32 = 17
------------
iters:9500
Loss:1.0308699028294888
Pred:[1 0 0 1 0 0 0 1]
True:[1 1 0 0 0 1 1 1]
87 + 112 = 145
------------
iters:9600
Loss:0.8116470094265957
Pred:[1 1 1 0 1 1 1 0]
True:[0 1 1 0 0 0 1 0]
16 + 82 = 238
------------
iters:9700
Loss:0.8516728701982583
Pred:[1 0 1 0 1 0 0 0]
True:[0 0 1 1 1 1 0 0]
46 + 14 = 168
------------
iters:9800
Loss:0.8313245053782707
Pred:[0 1 1 1 0 1 0 1]
True:[0 0 1 1 0 1 1 1]
31 + 24 = 117
------------
iters:9900
Loss:0.7476966153344927
Pred:[1 1 0 1 1 1 0 1]
True:[1 1 0 0 0 0 1 1]
107 + 88 = 221
------------
```


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2000
4000
6000
8000
10000