2020・06・03　ゼミ資料

ゼロから作るDeep Learning
1章～2章

まさぁぁ

1.1～1.4.1省略！

1章 クラス

class… 新しいデータの型を作るときに必要なやつ
データ構造を定義する時とかに有効

```
class クラス名:
    def __init__(self,引数1,引数2): # コンストラクタ
        # ここに初期化するときに必要なことを書く

    def メゾット名1(self,引数1,引数2): # メゾット1
        # 追加したい機能を書く

    def メゾット名2(self,引数1,引数2): # メゾット2
        # 追加したい機能を書く
```

1章 クラス

累積和クラスを書いてみよう

aの累積和を求めて

calc(i, j) で $\sum_{k=i}^{j-1} a_k$ を求める. O(1)

```
class Cum: # 累積和クラス
    def __init__(self,a):
        # ここに初期化するときに必要なことを書く



    def calc(self,i,j): # [i,j)の和を返す.
        return
```

1章 クラス

aの累積和を求めて

calc(i, j) で $\sum_{k=i}^{j-1} a_k$ を求める. O(1)

```
class Cum: # 累積和クラス
    def __init__(self,a):
        n = len(a)
        self.cum = [0] * (n + 1)
        for i in range(1, n + 1):
            self.cum[i] =a [i - 1]+self.cum[i - 1]

    def calc(self,i,j): # [i,j)の和を返す.
        return self.cum[j] - self.cum[i]
```

1章 Numpy, matplotlib

numpy, Matplotlib : 自分で読んで！

躓きそうなやつだけ貼っておく.

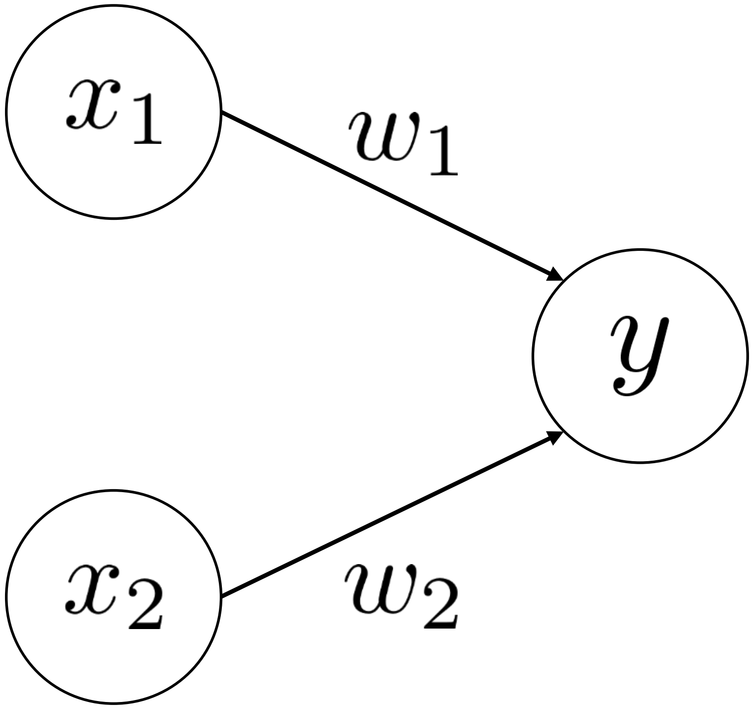
lena.pngはワークスペースのdataset_deepに保存した

教師なしと要領は同じ

```
import matplotlib.pyplot as plt
from matplotlib.image import imread
img = imread("dataset_deep/lena.png")
# 画像の読み込み ワークスペースのdataset_deep -> lena.png
plt.imshow(img)
plt.show()
```

2章：パーセプトロン

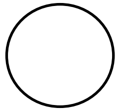
2章 パーセプトロン



x_1, x_2 : 入力信号

y : 出力信号

w_1, w_2 : 重み

 : ニューロン, ノード

2章 パーセプトロン

パーセプトロン

x_1 と x_2 の重み付きの和が
閾値 θ より大きい (またその時のみ)
ニューロンが発火する.

数式化

$$y = \begin{cases} 0 & (w_1 x_1 + w_2 x_2 \leq \theta) \\ 1 & (w_1 x_1 + w_2 x_2 > \theta) \end{cases}$$

実際に書いてみる

```
def per(x1,x2,w1,w2,theta):
    return w1*x1+w2*x2 > theta
```

2章 パーセプトロン

ANDゲートを作る

x_2 \ x_1	0	1
0	0	0
1	0	1

左の表を満たすように (w_1, w_2, θ) を定める

$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

$(w_1, w_2, \theta) = (0.5, 0.5, 0.7)$ でうまくいく.

```
def per(x1,x2,w1,w2,theta):
    return w1*x1+w2*x2 > theta

# AND回路
def AND(x1,x2):
    return per(x1,x2,0.5,0.5,0.7)
```

2章 パーセプトロン

Numpyに書き換える

$b = -\theta$ とする. (b をバイアスという)

$$y = \begin{cases} 0 & (b + w_1 x_1 + w_2 x_2 \leq 0) \\ 1 & (b + w_1 x_1 + w_2 x_2 > 0) \end{cases}$$

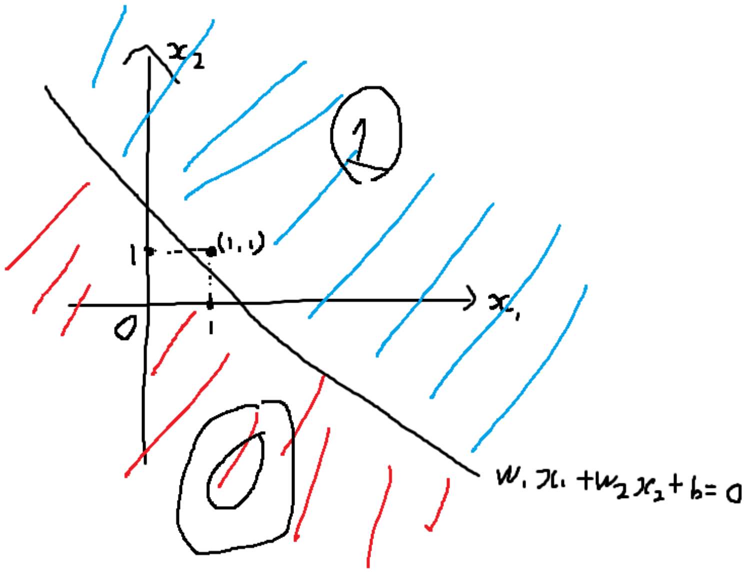
```
import numpy as np
def per(x1,x2,w1,w2,b): # w:重み b:バイアス
    x=np.array([x1,x2])
    w=np.array([w1,w2])
    return b+sum(x*w)>0

# AND回路
def AND(x1,x2):
    return per(x1,x2,0.5,0.5,-0.7)
```

2章 パーセプトロン

パーセプトロンの直感的理解

$$
y = \begin{cases} 0 & (b + w_1 x_1 + w_2 x_2 \leq 0) \\ 1 & (b + w_1 x_1 + w_2 x_2 > 0) \end{cases}
$$

便宜上 $(w_1, w_2, b) = (0.5, 0.5, -0.7)$ としている



2章 パーセプトロン

NANDゲートを作る

左の表を満たすように (w_1, w_2, b) を定める

$(w_1, w_2, b) = (-0.5, -0.5, 0.7)$ でうまくいく.

$x_2 \backslash x_1$	0	1
0	1	1
1	1	0

$$y = \begin{cases} 0 & (b + w_1 x_1 + w_2 x_2 \leq 0) \\ 1 & (b + w_1 x_1 + w_2 x_2 > 0) \end{cases}$$

```python
import numpy as np
def per(x1,x2,w1,w2,b): # w:重み b:バイアス
    x=np.array([x1,x2])
    w=np.array([w1,w2])
    return b+sum(x*w)>0

# NAND回路
def NAND(x1,x2):
    return per(x1,x2,-0.5,-0.5,0.7)
```

2章 パーセプトロン

NANDゲートを作る

左の表を満たすように (w_1, w_2, b) を定める

$(w_1, w_2, b) = (0.5, 0.5, -0.3)$ でうまくいく.

$x_2 \backslash x_1$	0	1
0	0	1
1	1	1

$$y = \begin{cases} 0 & (b + w_1x_1 + w_2x_2 \leq 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases}$$

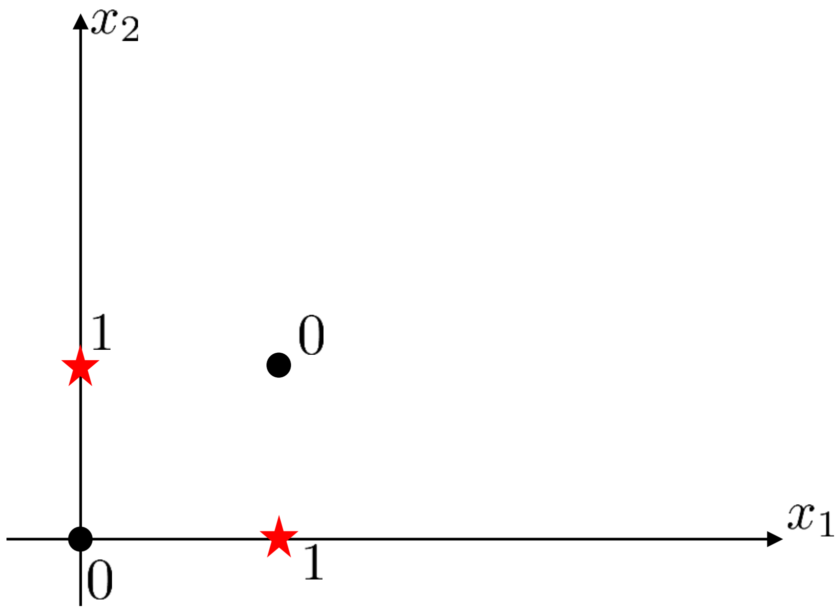
```
import numpy as np
def per(x1,x2,w1,w2,b): # w:重み b:バイアス
    x=np.array([x1,x2])
    w=np.array([w1,w2])
    return b+sum(x*w)>0

# or回路
def OR(x1,x2):
    return per(x1,x2,0.5,0.5,-0.3)
```

2章 パーセプトロン

XORゲートを作る

左の表を満たすように (w_1, w_2, b) を定める

これを満たす (w_1, w_2, b) は存在しない.



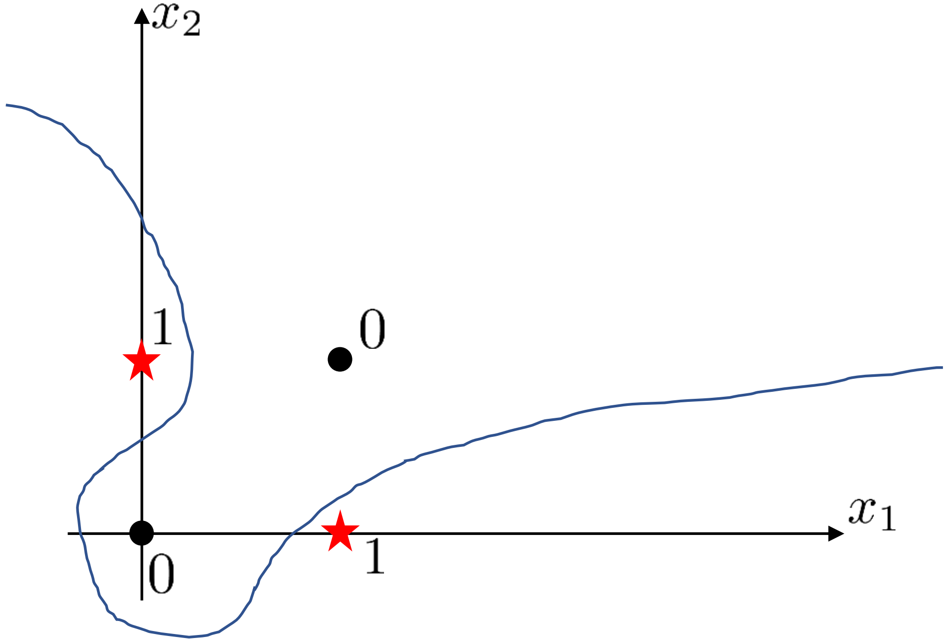
x_2 \ x_1	0	1
0	0	1
1	1	0

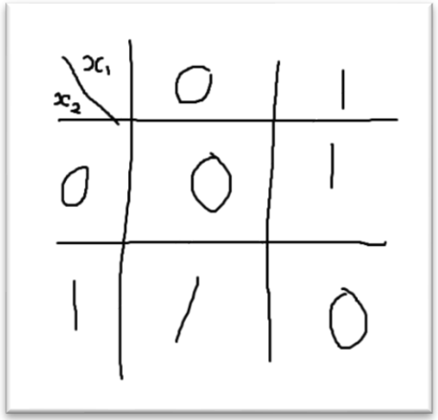
$$y = \begin{cases} 0 & (b + w_1 x_1 + w_2 x_2 \leq 0) \\ 1 & (b + w_1 x_1 + w_2 x_2 > 0) \end{cases}$$

2章 パーセプトロン

XORゲートを作る

1次元だと非線形な領域でしか実現不可能





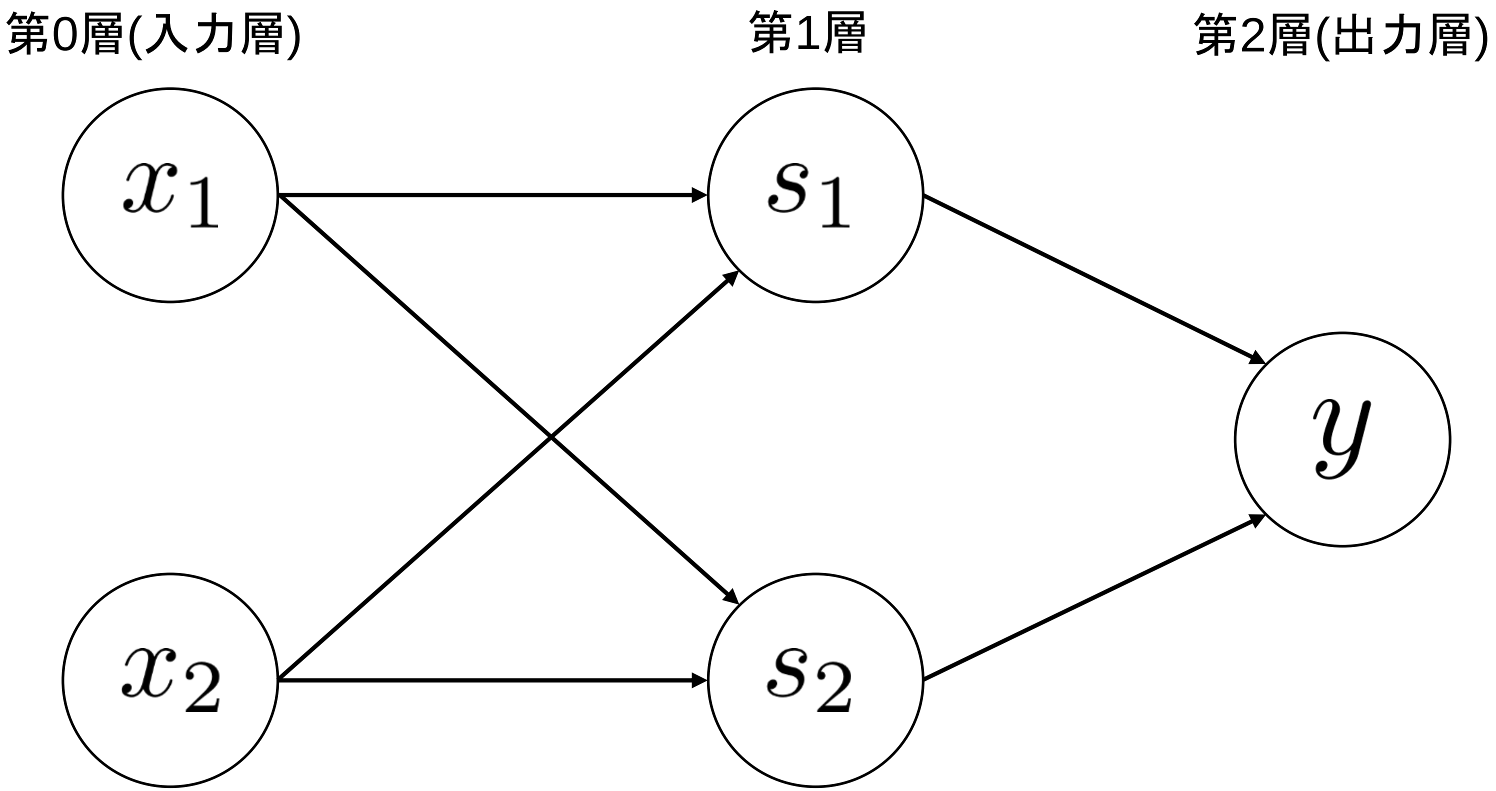
$$y = \begin{cases} 0 & (b + w_1x_1 + w_2x_2 \leq 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases}$$

→ 高次元(今までの組み合わせ)で考えてみる

2章 パーセプトロン

多層パーセプトロン

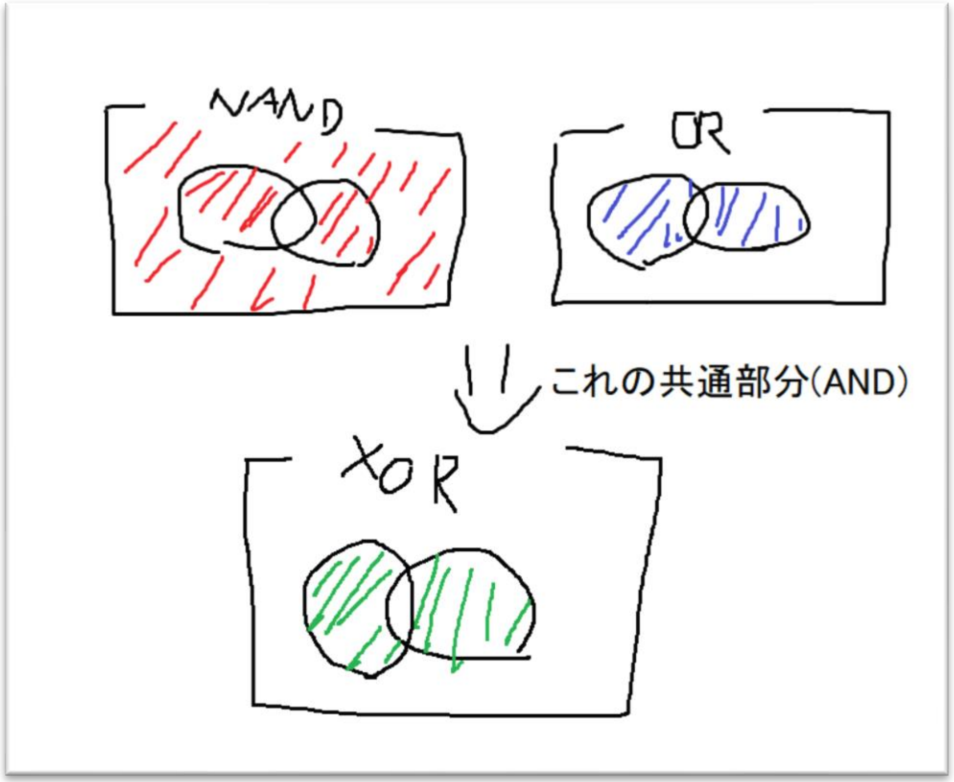
1次元だと実現不可能なものを層を重ねることで実現する



2章 パーセプトロン

XORをつくる

$a\ XOR\ b = (a\ NAND\ b)\ AND\ (a\ OR\ b)$ である.




```
# xor回路
def XOR(x1,x2):
    return AND(NAND(x1,x2),OR(x1,x2))
```

2章 パーセプトロン

まとめ

- 閾値を超えたら発火するパーセプトロンを学んだ.
- ANDゲートやORゲートは2層で実現可能
- 単層のパーセプトロンは線形領域しか表現できない.
- 多層パーセプトロンを使って非線形領域を表現することができる.
- 今回は省略するが多層パーセプトロンでコンピュータを表現できる.