

# パーセプトロン

前田利之（まえだ としゆき）

aipr@e-chan.jp

阪南大学 経営情報学部

(2022.10.14)

# ニューラルネットワークとは

- 人間の  を模倣したもの
- 知識を学習して、それを活用できる仕組み
- 今をときめく

,

の原理

- 上を含めた、現在の

,

ブームの牽引役

# ニューラルネットワークとは

- 人間の **脳の機構** を模倣したもの
- 知識を学習して、それを活用できる仕組み
- 今をときめく

の原理

- 上を含めた、現在の

ブームの牽引役

# ニューラルネットワークとは

- 人間の **脳の機構** を模倣したもの
- 知識を学習して、それを活用できる仕組み
- 今をときめく

**ディープラーニング (Deep Learning)**,

の原理

- 上を含めた、現在の

ブームの牽引役

# ニューラルネットワークとは

- 人間の **脳の機構** を模倣したもの
- 知識を学習して、それを活用できる仕組み
- 今をときめく **ディープラーニング (Deep Learning)**, **深層学習** の原理
- 上を含めた、現在の  
,  
**ブームの牽引役**

# ニューラルネットワークとは

- 人間の **脳の機構** を模倣したもの
- 知識を学習して、それを活用できる仕組み
- 今をときめく  
**ディープラーニング (Deep Learning)**,  
**深層学習** の原理
- 上を含めた、現在の  
**AI (Artificial Intelligence)**,  
 ブームの牽引役

# ニューラルネットワークとは

- 人間の **脳の機構** を模倣したもの
- 知識を学習して、それを活用できる仕組み
- 今をときめく  
**ディープラーニング (Deep Learning)**,  
**深層学習** の原理
- 上を含めた、現在の  
**AI (Artificial Intelligence)**,  
**人工知能** ブームの牽引役

# ニューラルネットワークとは（続き）

- 人間の脳の中には  という神経細胞がたくさんある（千億個のオーダー）
- 各  が  と呼ばれる接合部位によって繋がっている
- は入力される電気信号の合計が、ある一定の量（閾値）を超えると  し、シナプスによって次のニューロンに電気信号を出力する
- この動作の連続により、脳は信号の伝達を行っている

# ニューラルネットワークとは（続き）

- 人間の脳の中には **ニューロン** という神経細胞がたくさんある（千億個のオーダー）
- 各  が  と呼ばれる接合部位によって繋がっている
- は入力される電気信号の合計が、ある一定の量（閾値）を超えると  し、シナプスによって次のニューロンに電気信号を出力する
- この動作の連続により、脳は信号の伝達を行っている

# ニューラルネットワークとは（続き）

- 人間の脳の中には **ニューロン** という神経細胞がたくさんある（千億個のオーダー）
- 各 **ニューロン** が  と呼ばれる接合部位によって繋がっている
- は入力される電気信号の合計が、ある一定の量（閾値）を超えると  し、シナプスによって次のニューロンに電気信号を出力する
- この動作の連続により、脳は信号の伝達を行っている

# ニューラルネットワークとは（続き）

- 人間の脳の中には **ニューロン** という神経細胞がたくさんある（千億個のオーダー）
- 各 **ニューロン** が **シナプス** と呼ばれる接合部位によって繋がっている
- は入力される電気信号の合計が、ある一定の量（閾値）を超えると  し、シナプスによって次のニューロンに電気信号を出力する
- この動作の連続により、脳は信号の伝達を行っている

# ニューラルネットワークとは（続き）

- 人間の脳の中には **ニューロン** という神経細胞がたくさんある（千億個のオーダー）
- 各 **ニューロン** が **シナプス** と呼ばれる接合部位によって繋がっている
- **ニューロン** は入力される電気信号の合計が、ある一定の量（閾値）を超えると  し、シナプスによって次のニューロンに電気信号を出力する
- この動作の連続により、脳は信号の伝達を行っている

# ニューラルネットワークとは（続き）

- 人間の脳の中には **ニューロン** という神経細胞がたくさんある（千億個のオーダー）
- 各 **ニューロン** が **シナプス** と呼ばれる接合部位によって繋がっている
- **ニューロン** は入力される電気信号の合計が、ある一定の量（閾値）を超えると **発火** し、シナプスによって次のニューロンに電気信号を出力する
- この動作の連続により、脳は信号の伝達を行っている

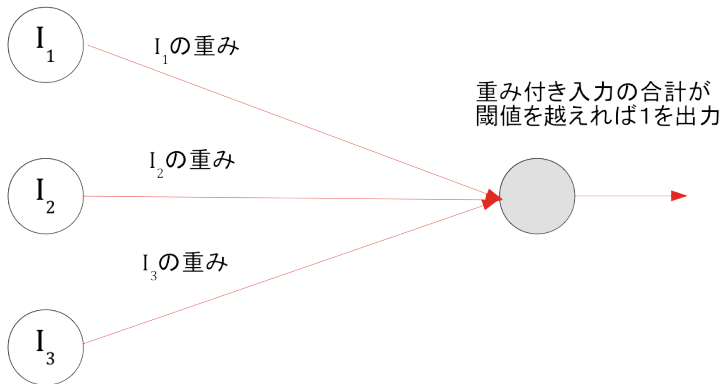
# ニューラルネットワーク（続き）

ニューラルネットシミュレーション（詳細は後述）

<https://puffin.hannan-u.ac.jp/lect/nsim/nnsim.html>

# パーセプトロン= (人工) ニューロン

入力



# パーセプトロンとは（続き）

$$y = f(u) = \begin{cases} 0 & (u < 0) \\ 1 & (u \geq 0) \end{cases}$$

$$u = \sum_{i=1}^n I_i w_i - \theta$$

- $I_i$  : 入力 (前の図では  $i=1,2,3$ )
- $w_i$  : それぞれの入力に対する重み
- $\theta$  : 閾値
- $y$  : 出力


# パーセプトロンとは（例）

- 出力は「夜に焼肉を食べに行く」で閾値は5
- $I_1$  は「ここ3日ほど肉を食べていない」で重みは5、
- $I_2$  は「朝ごはんを食べていない」で重みは3、
- $I_3$  は「最近疲れている」で重みは2、とする。

# パーセプトロンとは（例の続き）

- もし  $I_1$  が **1**（当たっている）なら、他が何であっても合計は5以上になるので出力は **1** となる（焼肉に行く）
- もし  $I_1$  が 0（最近肉を食べた）なら、
  - $I_2, I_3$  のどちらかだけが1の場合は合計は4以下なので出力は0となる（焼肉以外のものを食べる）
  - $I_2, I_3$  のどちらも1の場合は合計は5になり出力は1となる
- もし閾値を2に下げれば、3条件のいずれかが成り立てば出力は1となる

# パーセプトロンとは（論理ゲートの例）

- 論理ゲートが実現できれば、いろいろな  
（演算回路）を作りこむことが出来る  
（詳細略）

（屋上屋を重ねてる感じは否めない?!）

- 代表例：2入力・1出力の AND



- $I_1, I_2$  のどちらも1のときに限り出力は1
- ↑以外=どちらか（どちらも、も可）が0ならば0

# パーセプトロンとは（論理ゲートの例）

- 論理ゲートが実現できれば、いろいろな  
**計算機**（演算回路）を作りこむことが出来る  
（詳細略）

（屋上屋を重ねてる感じは否めない?!）

- 代表例：2入力・1出力の AND




- $I_1, I_2$  のどちらも1のときに限り出力は1
- ↑以外=どちらか（どちらも、も可）が0ならば0

# パーセプトロンとは（論理ゲートの例）

- 論理ゲートが実現できれば、いろいろな  
**計算機**（演算回路）を作りこむことが出来る  
（詳細略）  
（屋上屋を重ねてる感じは否めない?!）
- 代表例：2入力・1出力の AND **（論理積）**
  - $I_1, I_2$  のどちらも1のときに限り出力は1
  - ↑以外＝どちらか（どちらも、も可）が0ならば0

# パーセプトロンとは（論理ゲートの例）

- 2入力・1出力のパーセプトロンで AND を作るには…例えば  で実現できる（それ以外にも解はある）
  - このパラメータを変えることで他のゲートも出来る（後述）



# パーセプトロンとは（論理ゲートの例）

- 2入力・1出力のパーセプトロンで AND を作るには…例えば **重み 0.5, 閾値 0.8** で実現できる（それ以外にも解はある）
  - このパラメータを変えることで他のゲートも出来る（後述）



# パーセプトロンとは（論理ゲートの例）

- 2入力・1出力のパーセプトロンで AND を作るには…例えば **重み 0.5, 閾値 0.8** で実現できる（それ以外にも解はある）
  - このパラメータを変えることで他のゲートも出来る（後述）

パターン	$I_1$	$I_2$	$I_1 + I_2 - \theta$	出力
P1	0	0	-0.8	0
P2	0	1	-0.3	0
P3	1	0	-0.3	0
P4	1	1	0.2	1

## 【重要】実装の準備

- あえて別の名前にしない限り（デフォルトで）Colab ではグーグルドライブに **'Colab Notebooks'** というディレクトリ（フォルダ）を作るはずなので、それを前提とします。（自分で違う名前にした場合は、適宜読みかえてください）
- その下に実行コード `???.ipynb` が作られますので、今後の演習はそうにしてください。

## 【重要】実装の準備

- 'Colab Notebooks' の中に mymodules というフォルダを作って、モジュール（関数など）は、その中に ??? .py という名前で置いてください

[Google Drive]

```
...  
Colab Notebooks  
  and.ipynb  
  or.ipynb  
...  
mymodules  
  and_gate.py  
  or_gate.py  
...
```

# パーセプトロンとは（実装）

☆ 以下を mymodules の中に and\_gate.py という名前でファイルに保存する (Colab. ではなく TeraPad などのテキストエディタで書いてください)

```
# coding: utf-8
import numpy as np
def AND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = 0.8
    tmp = np.sum(w*x) - b
    if tmp <= 0:
        return 0
    else:
        return 1
```

# パーセプトロンとは（実行）

☆ 以下を Colab. 上で実行する（保存は and.ipynb という名前で）

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
bd = 'drive/My Drive/Colab Notebooks/mymodules'
import sys, os
sys.path.append(bd)
import and_gate as a
for xs in [(0, 0), (1, 0), (0, 1), (1, 1)]:
    y = a.AND(xs[0], xs[1])
    print(str(xs) + " -> " + str(y))
```

# 実装について

- 最初の5行はドライブ内のモジュールを使うおまじない(?)
- 最初だけ認証を求められるので、それに従ってURLに飛んでcopy&pasteして認証
- 参考URL (リンクがはってあります)
  - Google Colabotatory で Google Drive のファイルを読み込んでPythonで操作
  - 【備忘】 Google Colaboratory でカスタムモジュールをインポート

# パーセプトロンとは（実行）

## ☆ 実行結果

Mounted at /content/drive

(0, 0) → 0

(1, 0) → 0

(0, 1) → 0

(1, 1) → 1

# パーセプトロンとは（例）

## ■ 2入力・1出力のOR（論理積）

- $I_1, I_2$  のどちらも0ならば出力は0

- ↑以外＝どちらかが1のとき1

- ...  で実現できる（それ以外にも解はある）

## ■ 2入力・1出力のNAND（負論理積）

- $I_1, I_2$  のどちらも1ならば出力は0

- ↑以外＝どちらかが0のとき1

- ...  で実現できる（それ以外にも解はある）

# パーセプトロンとは（例）

- 2入力・1出力のOR（論理積）
  - $I_1, I_2$  のどちらも0ならば出力は0
  - ↑以外＝どちらかが1のとき1
  - ... **重み0.5, 閾値0.2** で実現できる（それ以外にも解はある）
- 2入力・1出力のNAND（負論理積）
  - $I_1, I_2$  のどちらも1ならば出力は0
  - ↑以外＝どちらかが0のとき1
  - ...  で実現できる（それ以外にも解はある）

# パーセプトロンとは（例）

- 2入力・1出力のOR（論理積）
  - $I_1, I_2$  のどちらも0ならば出力は0
  - ↑以外＝どちらかが1のとき1
  - ... **重み0.5, 閾値0.2** で実現できる（それ以外にも解はある）
- 2入力・1出力のNAND（負論理積）
  - $I_1, I_2$  のどちらも1ならば出力は0
  - ↑以外＝どちらかが0のとき1
  - ... **重み-0.5, 閾値0.8** で実現できる（それ以外にも解はある）

# パーセプトロンとは（実装）

☆ 以下を mymodules の中に or\_gate.py という名前でファイルに保存する

```
# coding: utf-8
import numpy as np
def OR(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = 0.2
    tmp = np.sum(w*x) - b
    if tmp <= 0:
        return 0
    else:
        return 1
```

# パーセプトロンとは（実装）

☆ 以下を mymodules の中に nand\_gate.py という名前でファイルに保存する

```
# coding: utf-8
import numpy as np
def NAND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([-0.5, -0.5])
    b = -0.8
    tmp = np.sum(w*x) - b
    if tmp <= 0:
        return 0
    else:
        return 1
```

# パーセプトロンとは（実行）

☆ 以下を Colab. 上で実行する（保存は or.ipynb という名前で）

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
bd = 'drive/My Drive/Colab Notebooks/mymodules'
import sys, os
sys.path.append(bd)
import or_gate as o
for xs in [(0, 0), (1, 0), (0, 1), (1, 1)]:
    y = o.OR(xs[0], xs[1])
    print(str(xs) + " -> " + str(y))
```

「結果」

```
(0, 0) -> 0
(1, 0) -> 1
(0, 1) -> 1
(1, 1) -> 1
```

# パーセプトロンとは（実行）

☆ 以下を Colab. 上で実行する（保存は nand.ipynb という名前で）

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
bd = 'drive/My Drive/Colab Notebooks/mymodules'
import sys, os
sys.path.append(bd)
import nand_gate as na
for xs in [(0, 0), (1, 0), (0, 1), (1, 1)]:
    y = na.NAND(xs[0], xs[1])
    print(str(xs) + " -> " + str(y))
```

「結果」

```
(0, 0) -> 1
(1, 0) -> 1
(0, 1) -> 1
(1, 1) -> 0
```

# パーセプトロンの限界

- 2入力・1出力の XOR (排他的論理積)
  - $I_1, I_2$  のどちらも 0、または、どちらも 1 ならば出力は 0
  - どちらかが 1、もう一方が 0 のとき 1
  - ... 実現でき
  - というのは  ので (詳細略)

# パーセプトロンの限界

- 2入力・1出力のXOR（排他的論理積）
  - $I_1, I_2$  のどちらも0、または、どちらも1ならば出力は0
  - どちらかが1、もう一方が0のとき1
  - ... 実現でき **ない！**
  - というのは  ので（詳細略）

# パーセプトロンの限界

- 2入力・1出力のXOR（排他的論理積）
  - $I_1, I_2$  のどちらも0、または、どちらも1ならば出力は0
  - どちらかが1、もう一方が0のとき1
  - ... 実現でき **ない！**
  - というのは **線形分離できない** ので（詳細略）

## 2次元空間 (OR)

☆論理和 (OR) は直線で、分離できる！

01

11

00

10

## 2次元空間 (OR)

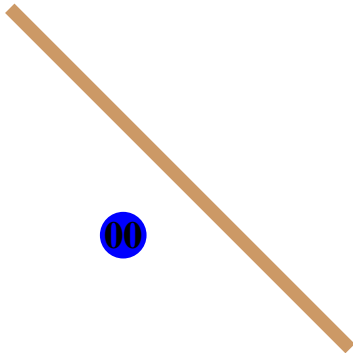
☆論理和 (OR) は直線で、分離できる！

01

11

00

10



## 2次元空間 (AND)

☆論理積 (AND) も直線で分離できる！

01

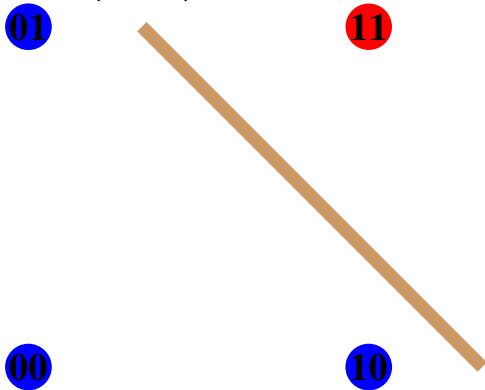
11

00

10

## 2次元空間 (AND)

☆論理積 (AND) も直線で分離できる！



## 2次元空間 (AND)

☆排他的論理和 (XOR) は直線で分離できない !!

01

11

00

10

## 2次元空間 (AND)

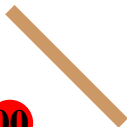
☆排他的論理和 (XOR) は直線で分離できない !!

01

11

00

10



## 2次元空間 (AND)

☆排他的論理和 (XOR) は直線で分離できない !!

01

11

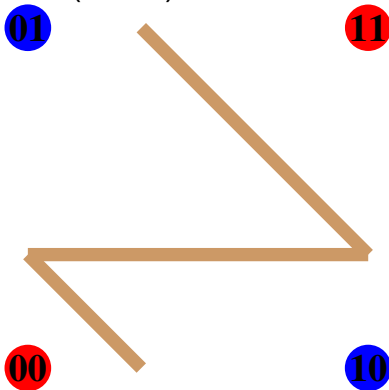
00

10



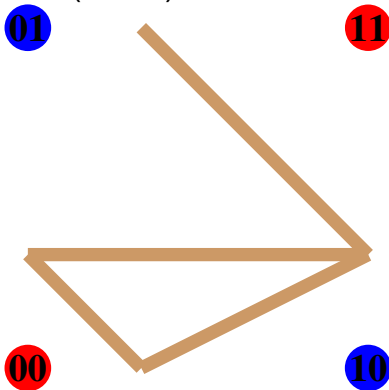
## 2次元空間 (AND)

☆排他的論理和 (XOR) は直線で分離できない !!



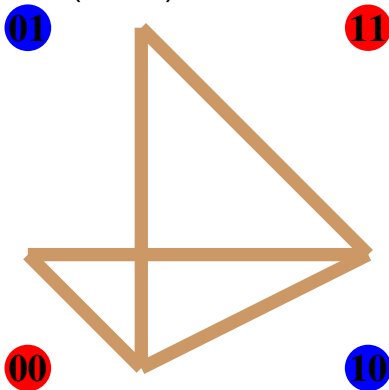
## 2次元空間 (AND)

☆排他的論理和 (XOR) は直線で分離できない !!



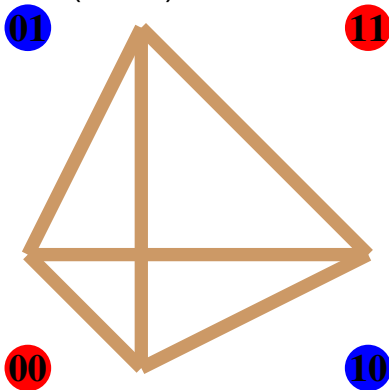
## 2次元空間 (AND)

☆排他的論理和 (XOR) は直線で分離できない !!



## 2次元空間 (AND)

☆排他的論理和 (XOR) は直線で分離できない !!



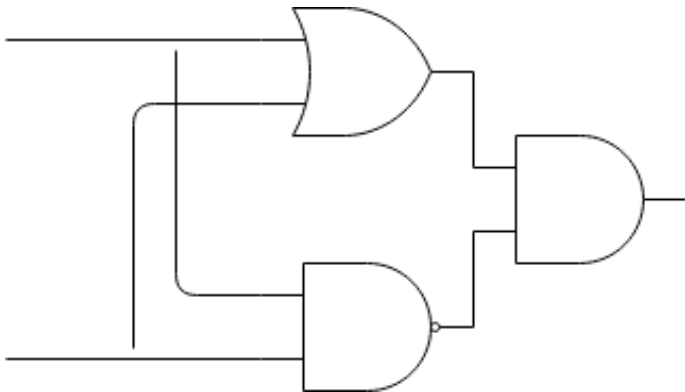
# パーセプトロンの限界

- 2入力・1出力の XOR（排他的論理積）が線形分離できなければ、どうするか？
- $\Rightarrow$  単純ではない =  により、実現できる

# パーセプトロンの限界

- 2入力・1出力の XOR（排他的論理積）が線形分離できなければ、どうするか？
- $\Rightarrow$  単純ではない = **組み合わせ（階層化）** により、実現できる

# パーセプトロンによる XOR



☆左下のゲートは気づいかかもしれませんが、出力に○が付いていて、負論理を表しています

# パーセプトロンとは（実装）

☆ 以下を mymodules の中に xor\_gate.py という名前でファイルに保存する

```
# coding: utf-8
from and_gate import AND
from or_gate import OR
from nand_gate import NAND
def XOR(x1, x2):
    s1 = NAND(x1, x2)
    s2 = OR(x1, x2)
    y = AND(s1, s2)
    return y
```

# パーセプトロンとは（実装）

☆ 以下を Colab. 上で実行する（保存は xor.ipynb という名前で）

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
bd = 'drive/My Drive/Colab Notebooks/mymodules'
import sys, os
sys.path.append(bd)
import xor_gate as x
for xs in [(0, 0), (1, 0), (0, 1), (1, 1)]:
    y = x.XOR(xs[0], xs[1])
    print(str(xs) + " -> " + str(y))
```

「結果」

```
(0, 0) -> 0
(1, 0) -> 1
(0, 1) -> 1
(1, 1) -> 0
```

# おしまい

質問・コメント等あれば、何でもお気軽に  
aipr@e-chan.jp に  
メールください！