at-2.ニューラルネットワークの基礎

（ディープラーニングのシステムとプログラミング）

（全１２回）

https://www.kkaneko.jp/ai/at/index.html

金子邦彦

CC BY NC SA

1. ニューラルネットワークの基礎を説明
2. 機械学習の目的、ニューラルネットワークの仕組み、学習の原理など
3. 畳み込みについても基本的な操作から応用まで学べる
4. 図、数式、サンプルコードを適所に交えることで、理論と実践の両面から理解を深める

アウトライン

１．機械学習

２．ニューラルネットワークの仕組み

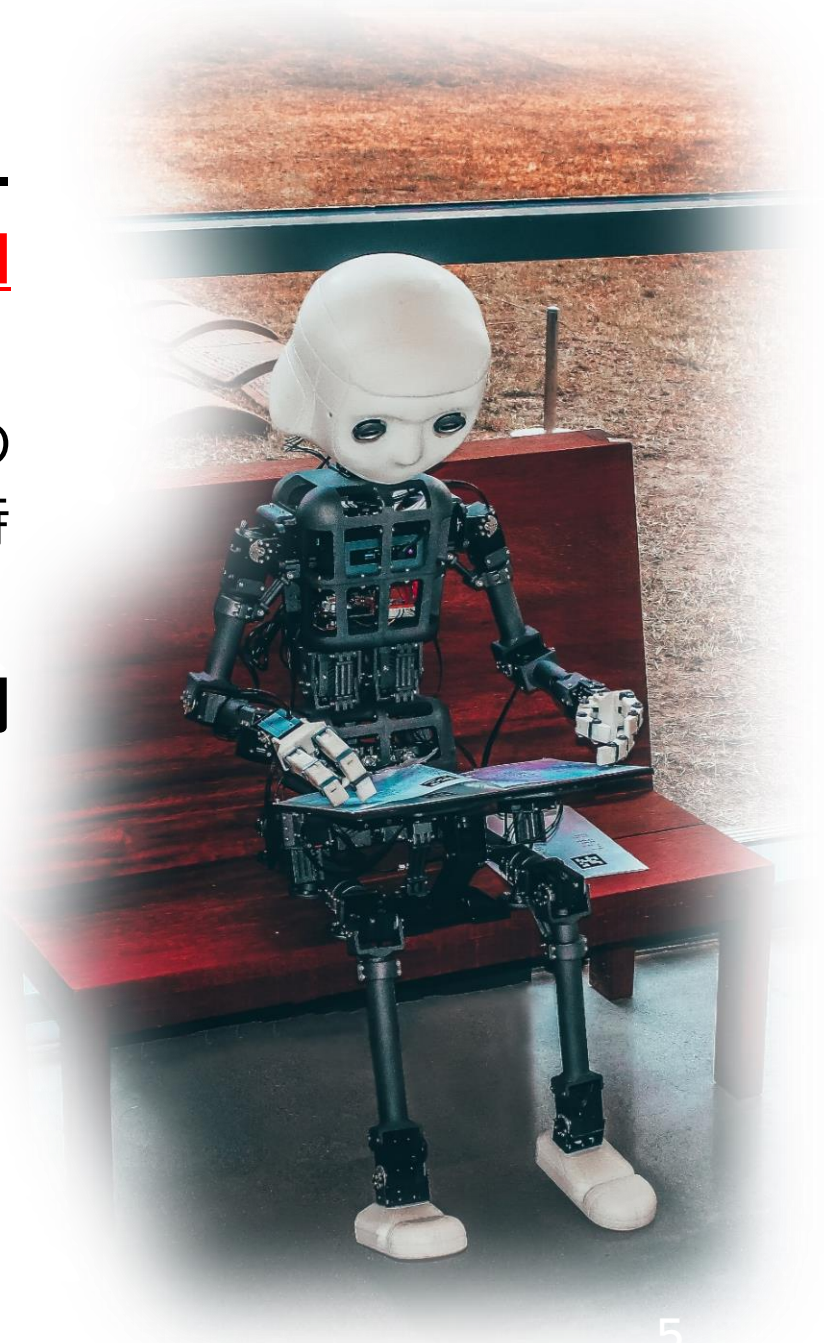
３．畳み込みの仕組み

４．画像での畳み込み

10-1. 機械学習

機械学習

- **機械学習**は，**コンピュータ**が**データ**を使用して**学習**することより**知的能力を向上**させる技術
- 情報の抽出能力，ルールや知識のプログラム化不要などの**特徴**を持つ
- 他の方法では**解決が難しかった問題に取り組むことが可能**に



機械学習が目標とする作業

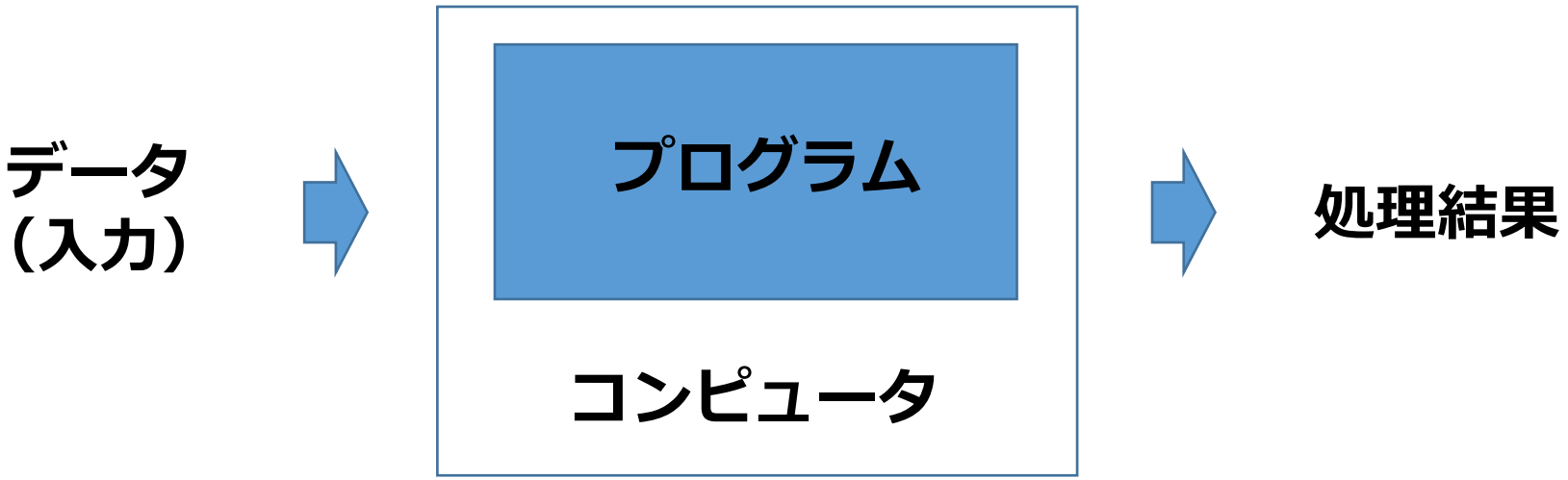
ルール化，プログラム化が難しい作業

- 直感
- 主観
- 経験

→　人間にたやすい
　　しかし，プログラム化は困難

① 一般のプログラミング

・プログラムは人間が作成し，テストし，調整する．



② 機械学習での予測

・学習による上達の能力

機械学習での汎化

訓練データ

入力	正解
9	500
11	500
12	1000
14	1000

訓練データの汎化

入力	予測結果
7	**500**
8	**500**
9	500
10	**500**
11	500
12	1000
13	**1000**
14	1000
15	**1000**
16	**1000**

汎化により，**未知のデータ**についても**予測**ができるようになる

- **汎化は１００%成功するわけでない.**
- **訓練データ**とは別のデータ（**検証データ**）を用いて検証する

「**汎化**は，プログラミングを補うもの」と
考えられるようにも

- ふつうのプログラミング：

あらゆる事態を想定して，プログラムを作成

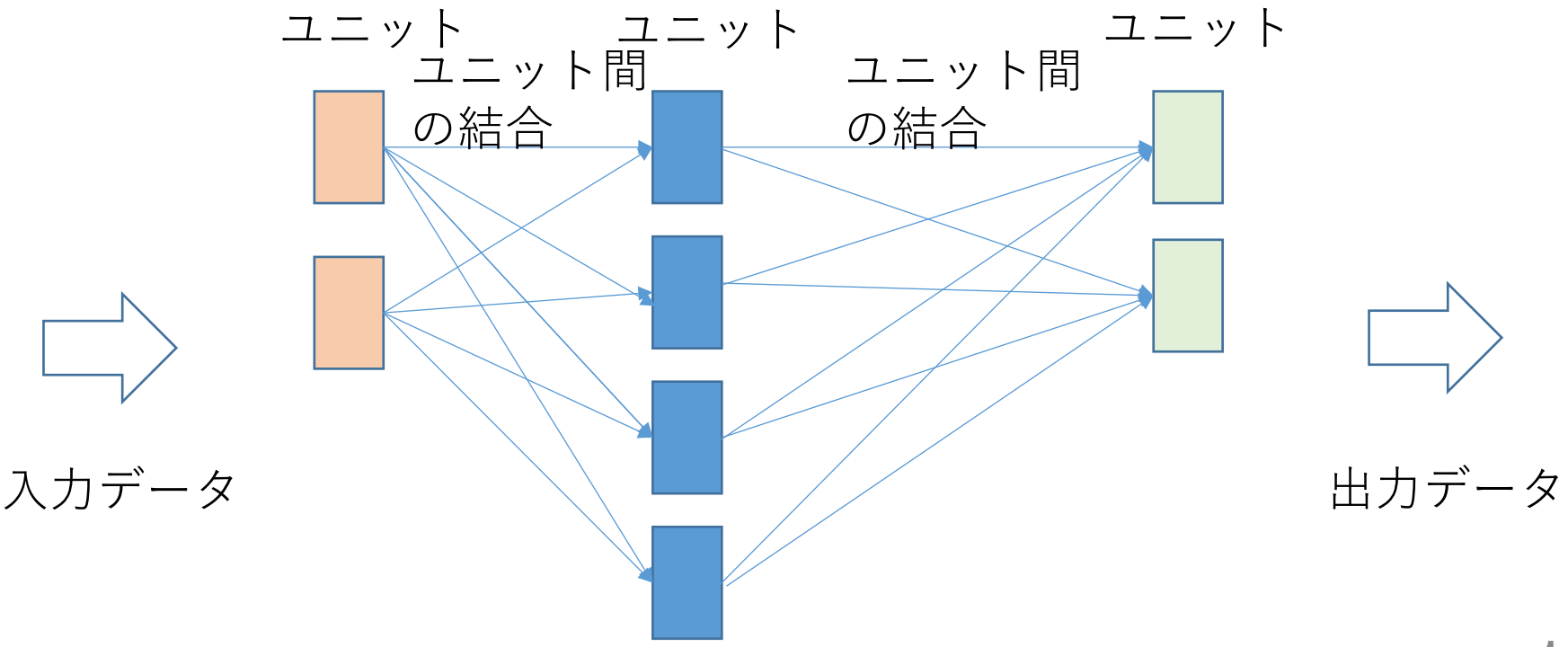
- 汎化：

未知のデータについても処理できる

10-2. ニューラルネットワークの仕組み

ニューラルネットワーク

- **機械学習の能力を持つ.**
- コンピュータで動作.
- **ユニット**がつながり，ニューラルネットネットワークを構成.



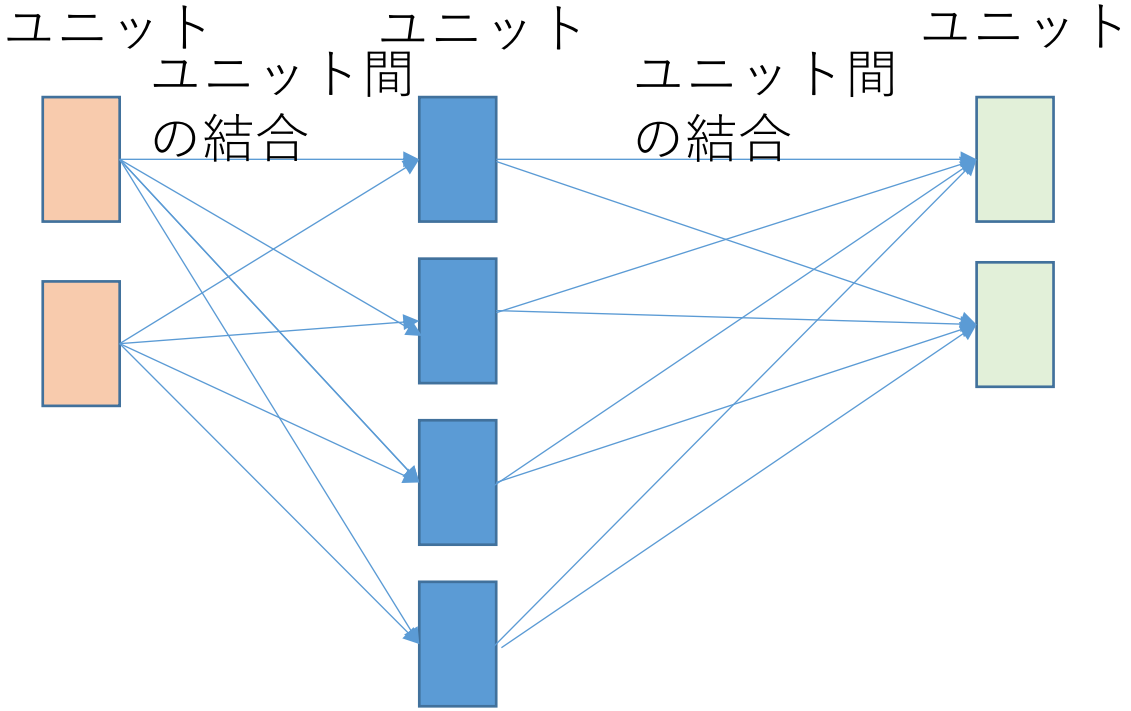
ニューラルネットワークの基礎

- **単純な仕組み．明確な原理に基づいて処理を行う**

ユニットは，基本，**入力の重みづけ**，**合計とバイアス**，**活性化関数の適用**を行う

- **ユニットのネットワークである**

多数の**ユニット**とそれらの**ユニット間の結合**から構成



入力の重みづけ，合計とバイアス，活性化関数の適用

1. **入力の重みづけ**: **ユニット**は**複数の入力を受け取り**、それぞれの入力に対して重みを掛ける（掛け算）
2. **合計とバイアス**: **1. の合計を求め，バイアスと呼ばれる値を足す（足し算）**
3. **活性化関数の適用**: 2. の値に，活性化関数を適用して出力値を得る）

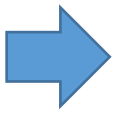
10.1 × 0.1 ⇒ 1.01

-0.5 × 0.8 ⇒ -0.4

0.2 × -0.5 ⇒ -0.1

入力 重み

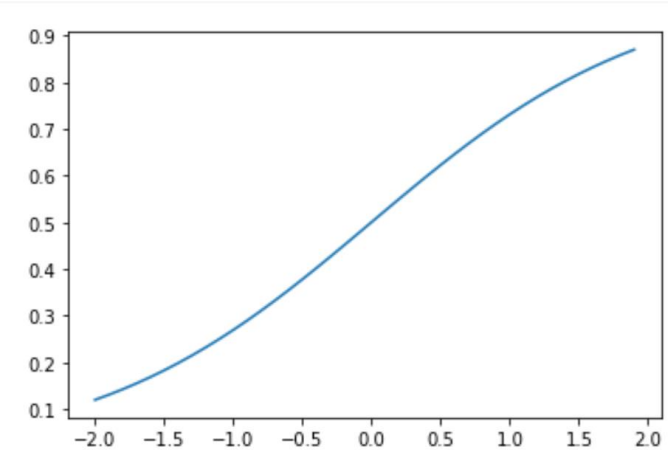
合計
0.51



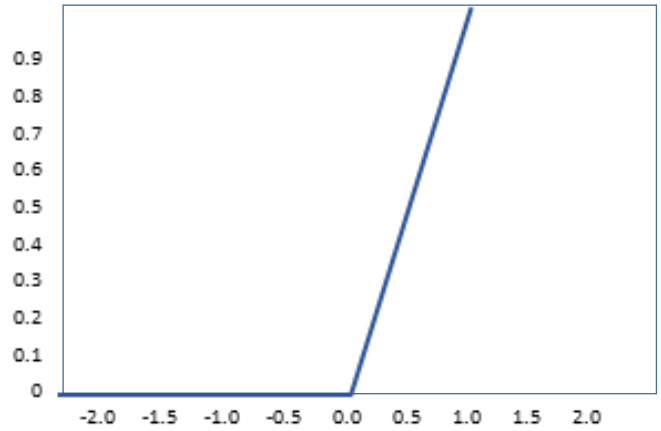
合計に
応じた出力値
0.51

ニューラルネットワークの活性化関数のバリエーション

○ **活性化関数**は**値を変換する機能**を持ち，**さまざまな種類**がある



シグモイド



ReLU
（2011年発表）

URL: https://trinket.io/python/af9e01d901

main.py

```
def relu(x):
  if x < 0:
    return 0
  else:
    return x

def n(x1, x2, x3):
  s = 0.1 * x1 + -0.5 * x2 + 0.2 * x3
  return relu(s)

x = -2
print("x = ", x, "leru(x) =", relu(x))
x = -1
print("x = ", x, "leru(x) =", relu(x))
x = 0
print("x = ", x, "leru(x) =", relu(x))
x = 1
print("x = ", x, "leru(x) =", relu(x))
x = 2
print("x = ", x, "leru(x) =", relu(x))

print("s(0.1, 0.8, -0.5) =", n(0.1, 0.8, -0.5))
print("s(5.1, 0.8, -0.5) =", n(5.1, 0.8, -0.5))
print("s(10.1, 0.8, -0.5) =", n(10.1, 0.8, -0.5))
print("s(15.1, 0.8, -0.5) =", n(15.1, 0.8, -0.5))
```

Result　Instructi

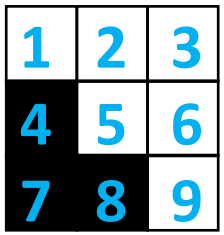
Powered by trinket

```
('x = ', -2, 'leru(x) =', 0)
('x = ', -1, 'leru(x) =', 0)
('x = ', 0, 'leru(x) =', 0)
('x = ', 1, 'leru(x) =', 1)
('x = ', 2, 'leru(x) =', 2)
('s(0.1, 0.8, -0.5) =', 0)
('s(5.1, 0.8, -0.5) =', 0.01)
('s(10.1, 0.8, -0.5) =', 0.51)
('s(15.1, 0.8, -0.5) =', 1.01)
```

s(20.1, 0.8, -0.5) の値は？ S(25.1, 0.8, -0.5) の値は？

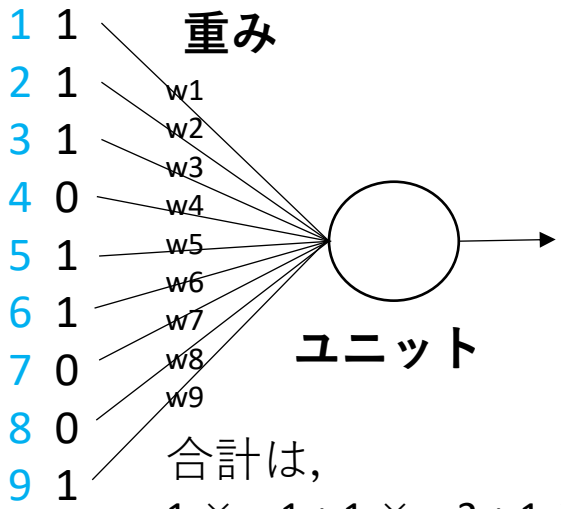
ニューラルネットワークの処理の原理

1. **入力の重みづけ**:
   1 × w1, 1 × w2, 1 × w3, 0 × w4, 1 × w5, 1 × w6, 0 × w7, 0 × w8, 1 × w9
   （w1 から w9 は重み）
2. **合計とバイアス**:
   1 × w1 + 1 × w2 + 1 × w3 + 0 × w4 + 1 × w5 + 1 × w6 + 0 × w7 + 0 × w8 + 1 × w9 + b （b は**バイアス．プラスや０やマイナスの数**）
3. **活性化関数の適用による出力値の取得**
   f(1 × w1 + 1 × w2 + 1 × w3 + 0 × w4 + 1 × w5 + 1 × w6 + 0 × w7 + 0 × w8 + 1 × w9 + b) （***f*** は**活性化関数**）



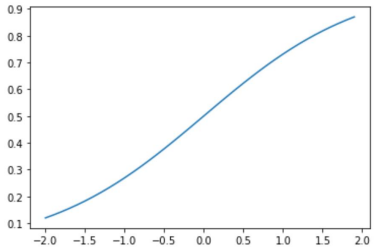
入力

白黒の画像
（画素は 0 または 1）

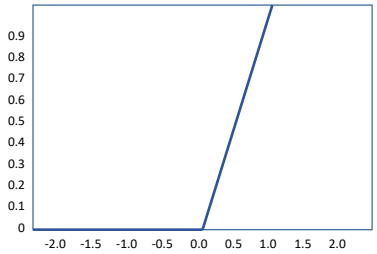


合計は，
1 × w1 + 1 × w2 + 1 × w3 +
0 × w4 + 1 × w5 + 1 × w6 +
0 × w7 + 0 × w8 + 1 × w9

活性化関数はさまざまな種類



シグモイド

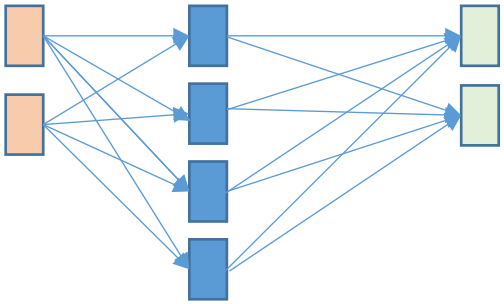


ReLU
（2011年発表）

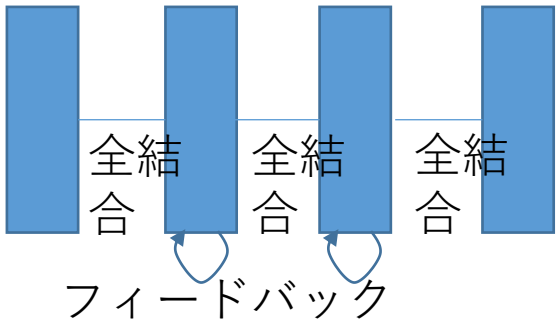
ニューラルネットワークの構造のバリエーション

○ **層構造で全結合のもの**が**最もシンプル**

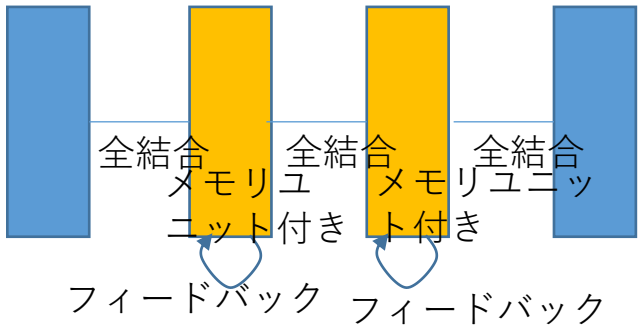
- **層構造**：**前の層**から結果を**受けとって**，**次の層**へ結果を**渡す**
- **全結合**：二層の間で，すべての**ユニット同士が結合**



○ フィードバック，メモリなど複雑なものもある

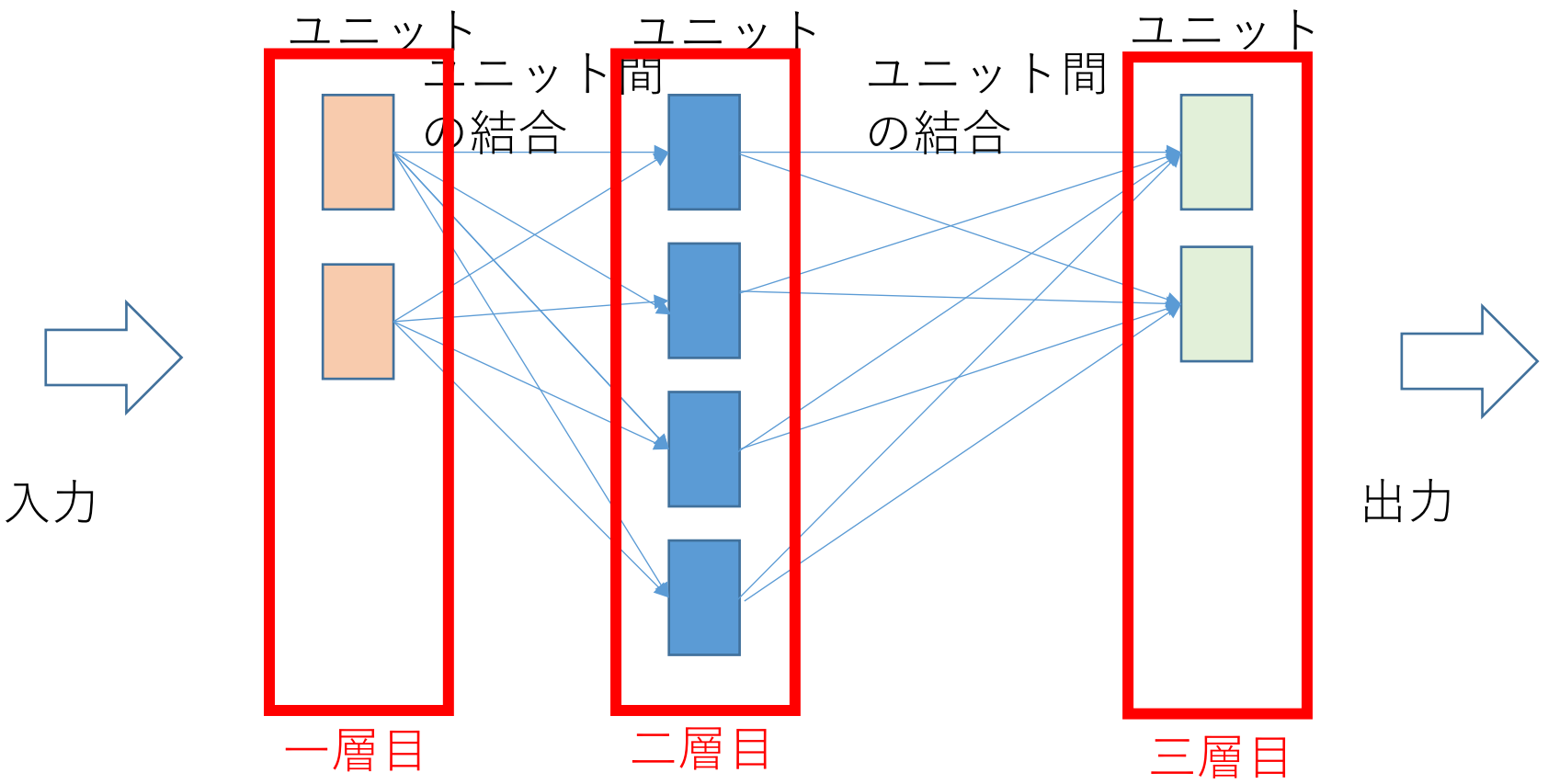


RNN



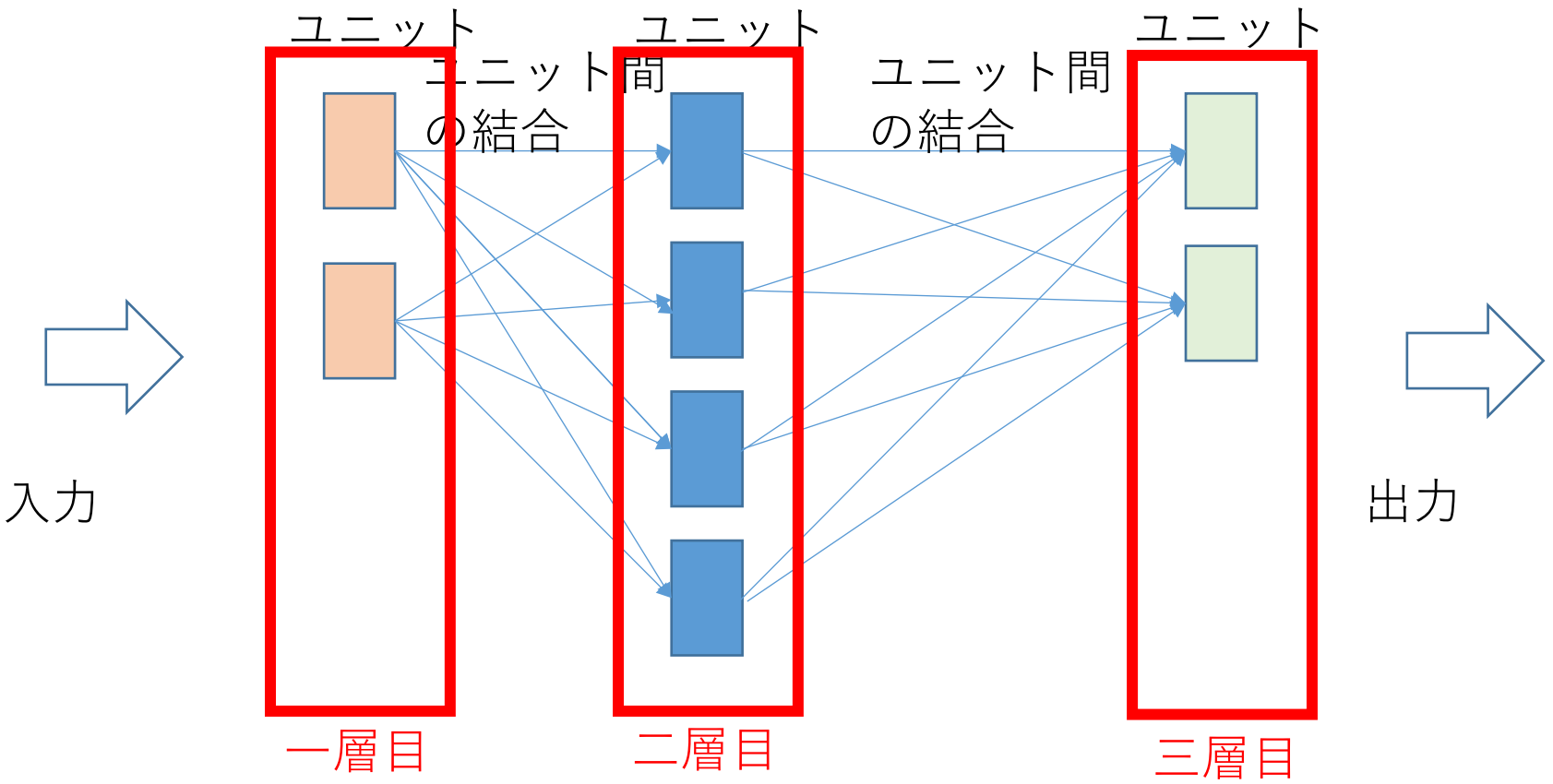
LSTM

層構造とデータの流れ



- **層構造**では，**データ**は，**入力から出力への一方向**に流れる
- **各層**は、**同じ種類のユニット**で構成

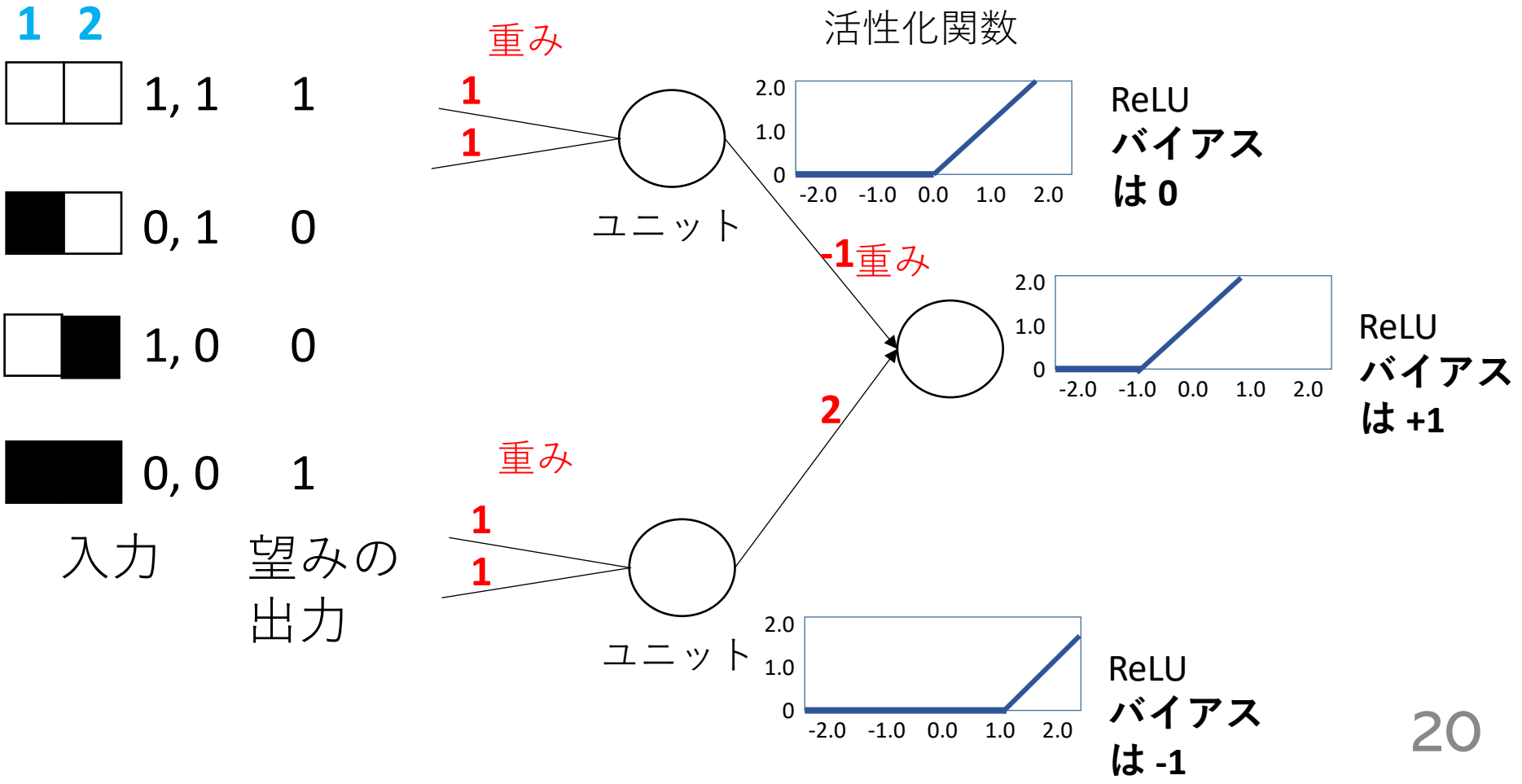
全結合のニューラルネットワーク



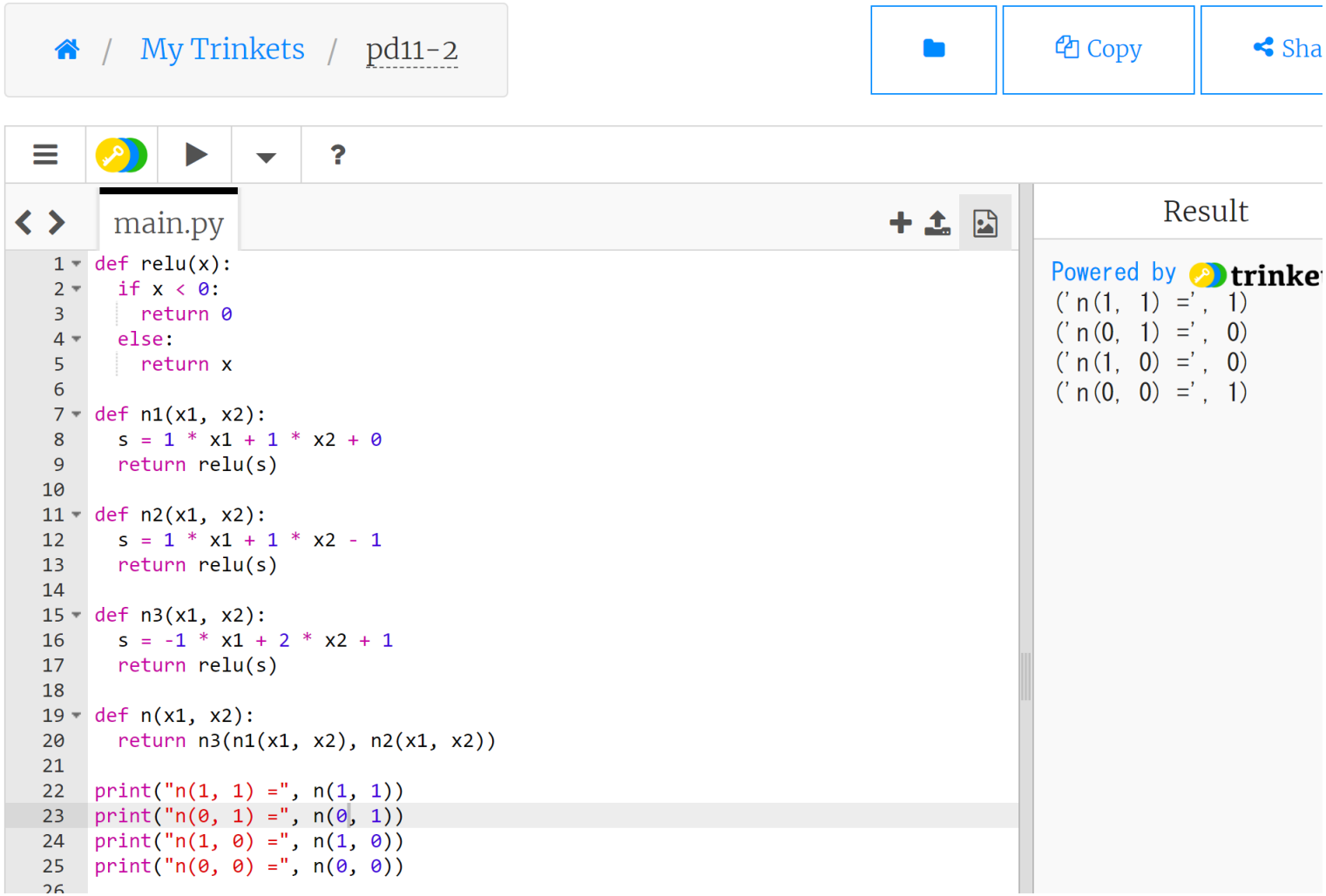
- **全結合**では，**２つの層の間**で，**すべてのユニット同士が結合**される
- **１つのユニットの出力**は，次の層の**すべてのユニットに影響**

ニューラルネットワークが多様に利用できる原理

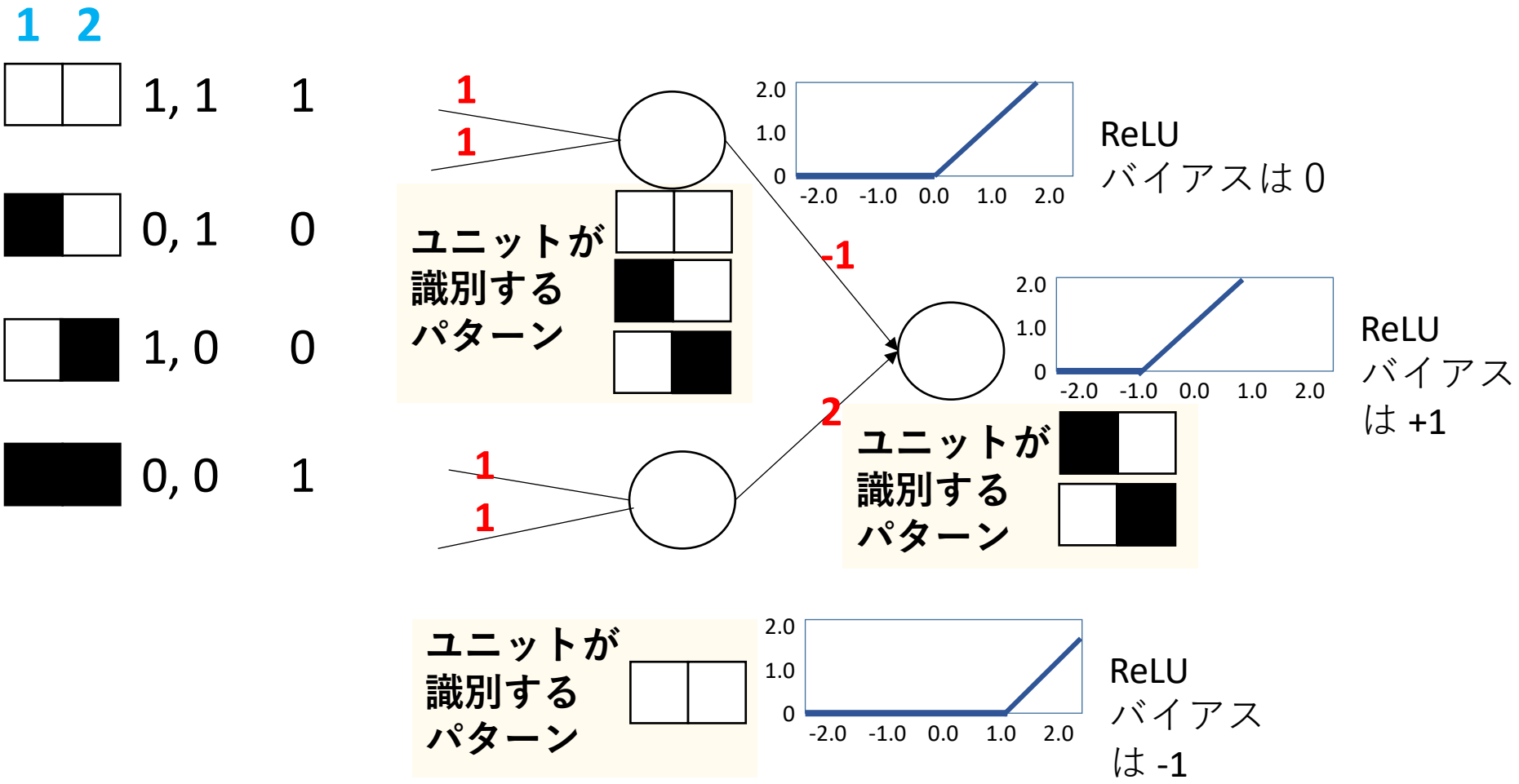
- **望みの出力が得らえる**ように，**重み，バイアスを調整**
- このとき，結合のさせ方，ユニット数，ユニットの種類などは**変化させない**



URL: https://trinket.io/python/8a0eed74c9



それぞれの**ユニット**が「**特定のパターンを識別している**」と考えることもできる



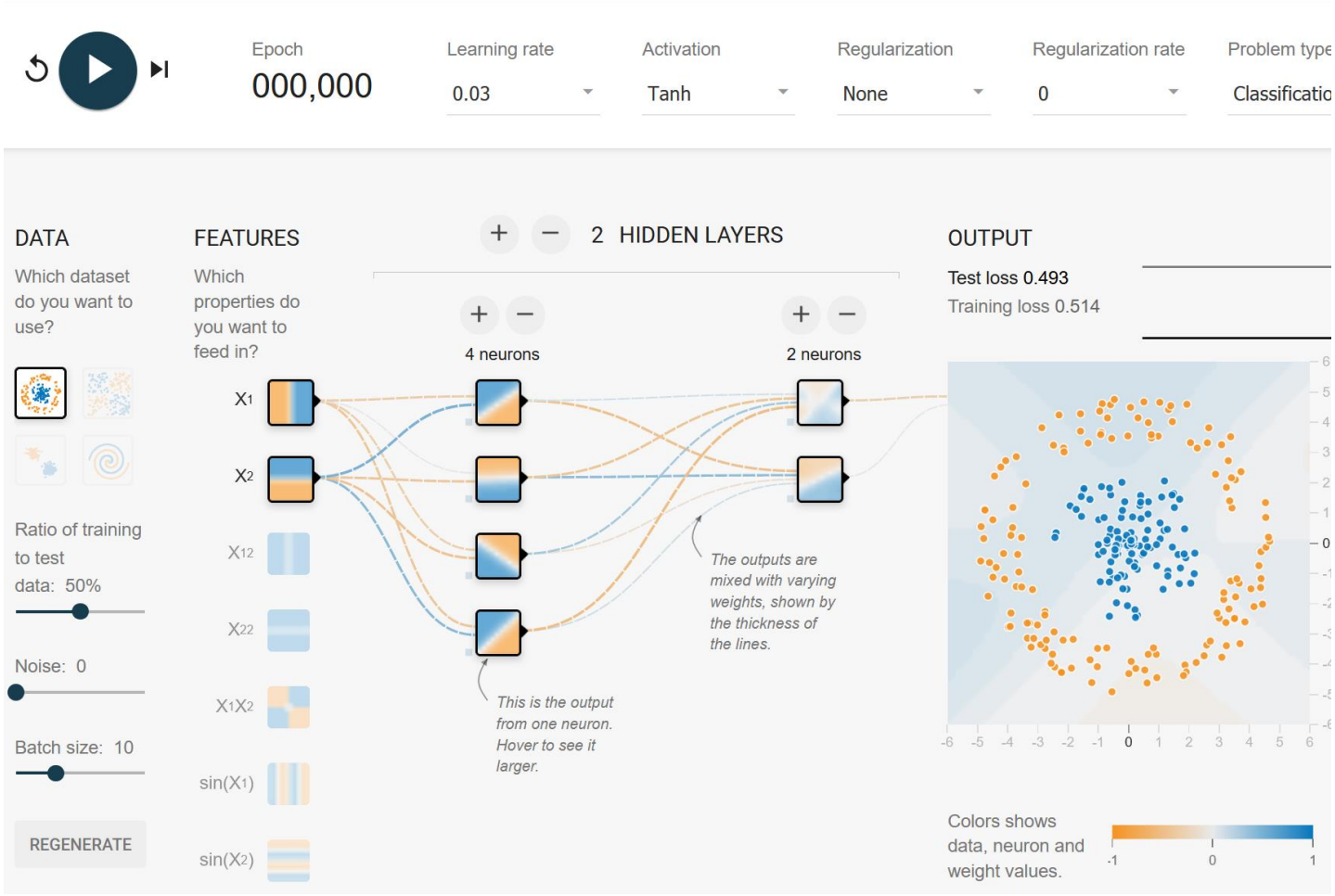
ニューラルネットワークの教師あり学習

- **訓練データ**（学習のためのデータ）を使用
- **学習**は**自動**で行われる

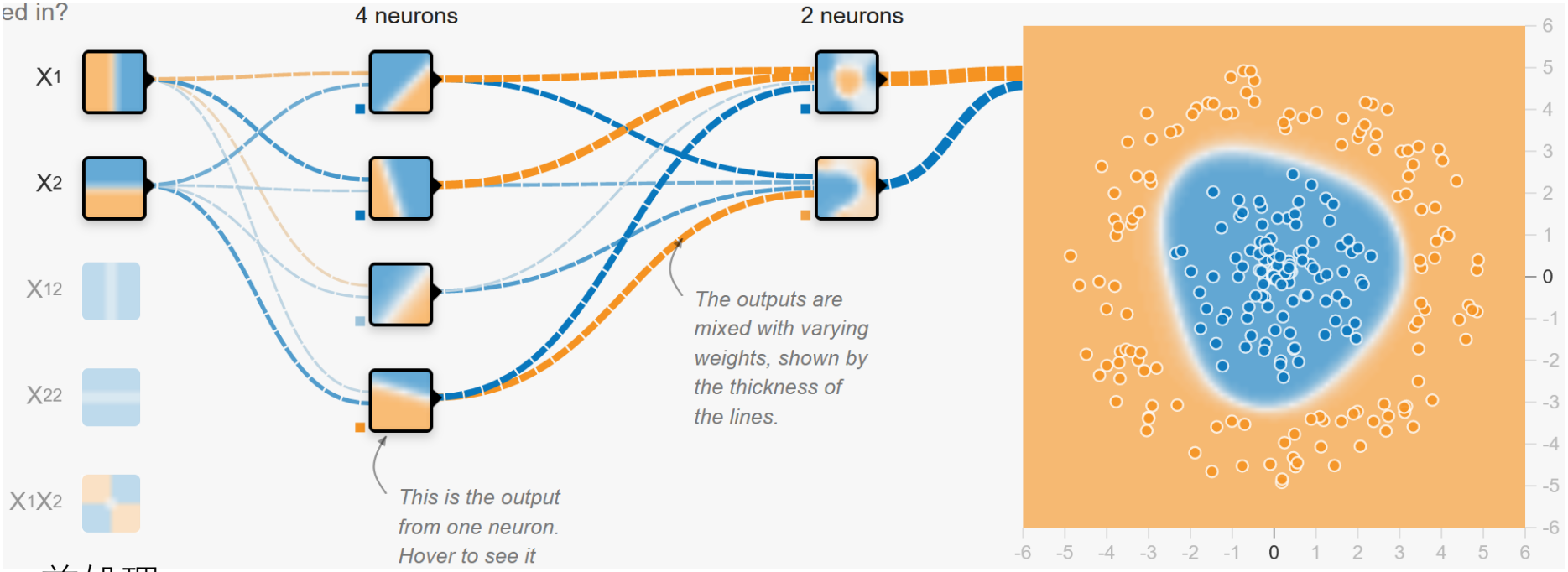
① **訓練データ**により，**ニューラルネット**から結果を出し，正解との**誤差**を得る

② **ユニット間の結合の重み**の**調整**により、**誤差**を減らす

学習能力をコンピュータに組み込んでおき，あとでデータを与えて学習させる



http://playground.tensorflow.org



前処理
（データが青い部分にあれば活性化）

データが中央にあれば活性化

→結合→ 1層目 →結合→ 2層目

ニューラルネットワーク

URL: https://playground.tensorflow.org/

ニューラルネットワークの学習

- **データ**（学習のためのデータ）を使用

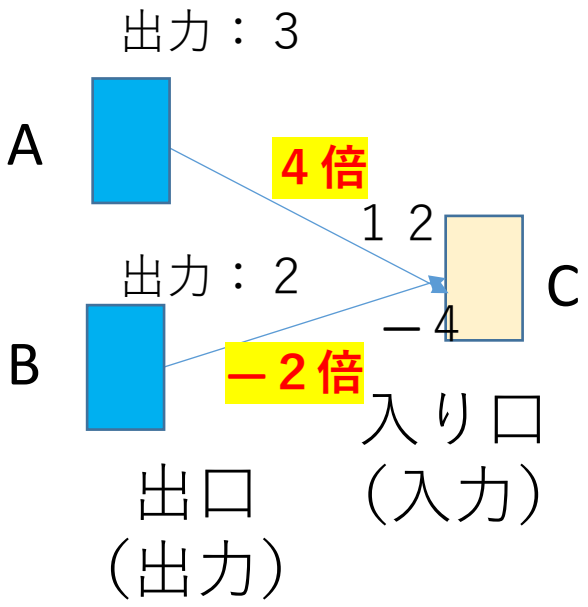
① **データ**により，**ニューラルネット**を動かし，**誤差**を得る

② **結合の重みとバイアスの調整**により、**誤差**を減らす

- ユニットの数が増えたり減ったりなどではない

- **誤差**が**減らなくなったら**、**最適**になったとみなす

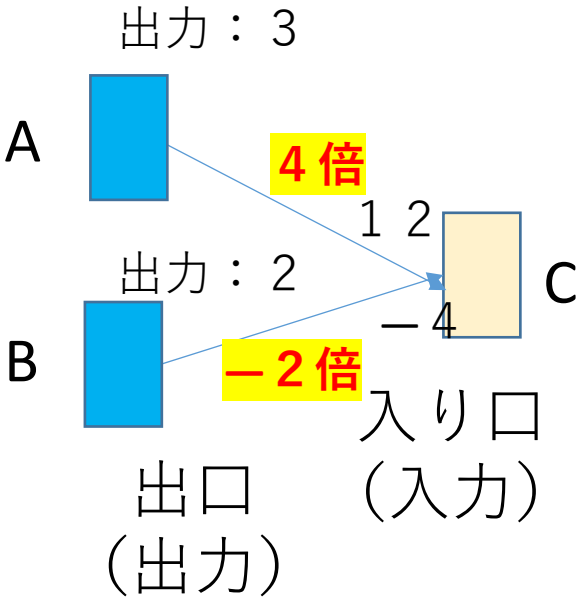
結合の重みの調整



ユニット間の結合の
　「○○**倍**」
は、学習の途中で変化する

ニューラルネットワークの**学習**は、
望み通りの出力が得られるように、「○○**倍**」のところ（**結合の重み**）を**調整**すること

確認問題

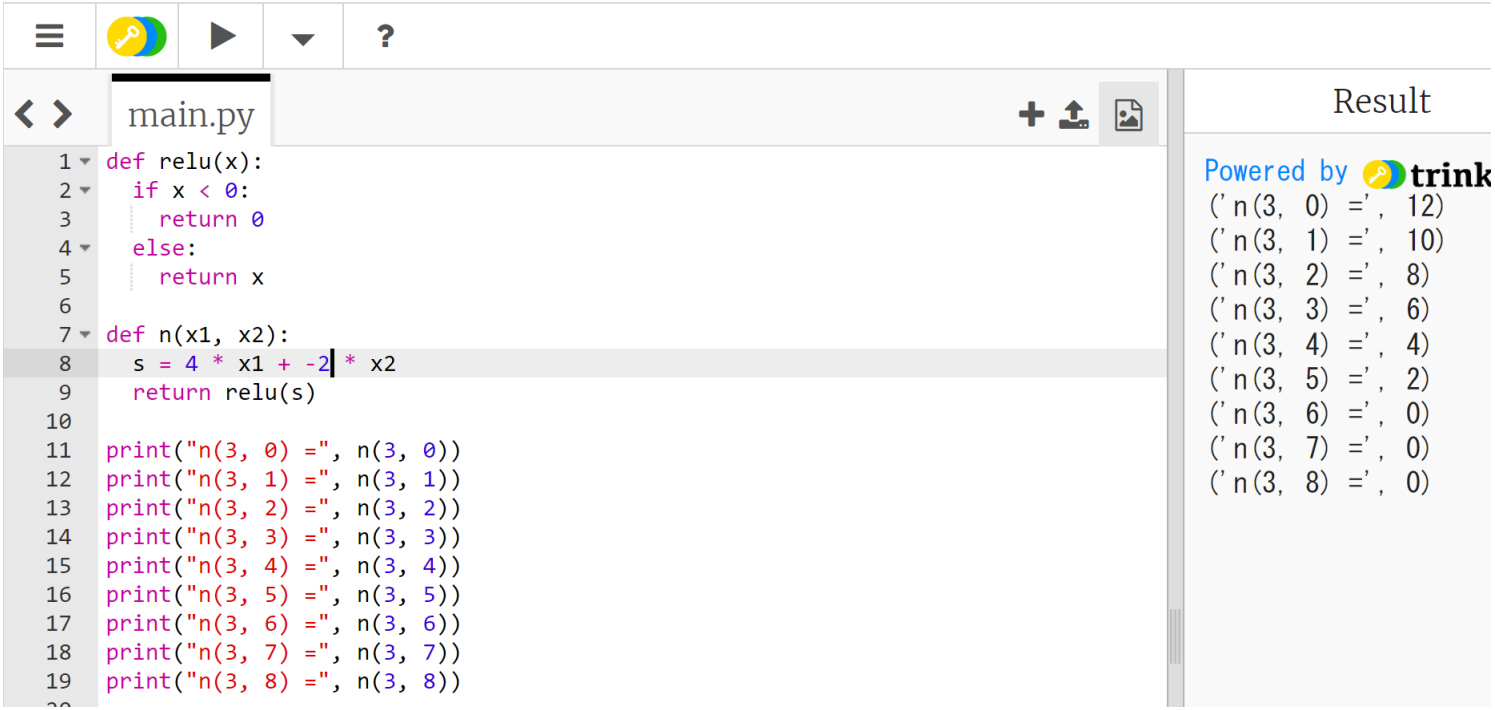


次のような**ユニット**がある

- 入力の合計が**0以上**のとき、**活性化**し、**1**を出力する
- 入力の合計が**0未満**のとき、**非活性化**し、**0**を出力する

ユニットAの出力は3であるとする。
ユニットBの出力が**2以下**のとき
　ユニットCは**活性化**
ユニットBの出力が少しでも**2を超えたとき**
　ユニットCは**非活性化**
にしたい。結合の重みをどう調整するか？
　（答え）「－2倍」を「－6倍」へ

URL: https://trinket.io/python/b1f04a9758



-2 を -6 に変えて、変化を見る

ニューラルネットワークのまとめ

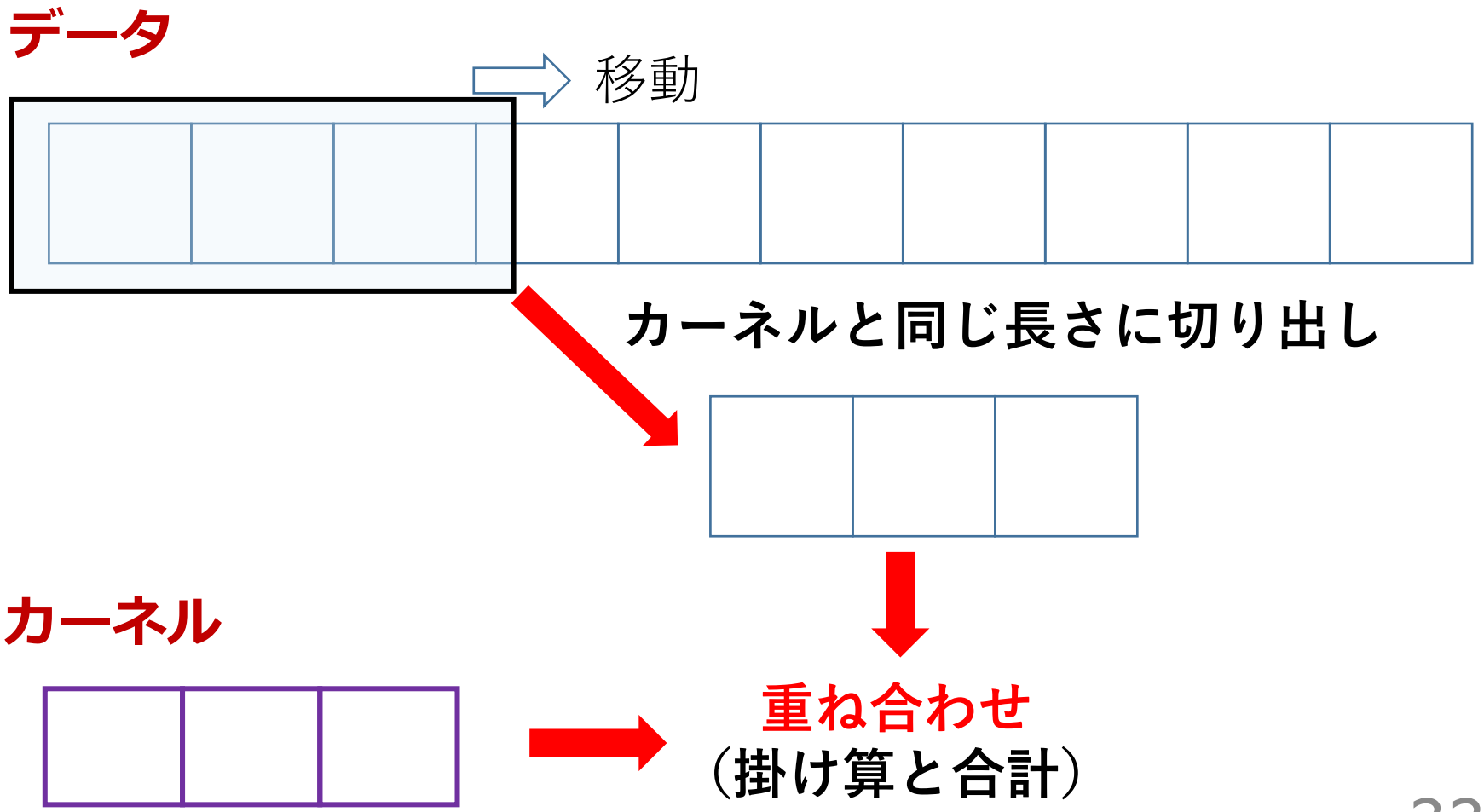
- **ニューラルネットワーク**は，単純な原理で動く
- **ユニット**は**複数の入力**を受け取る．中では，**活性化関数**を用いた**値の変換**を行う．
- **層構造**と**全結合**のニューラルネットワークが最もシンプルである．**入力から出力の方向へ一方向にデータが流れる**．**２つの層の間のユニット**は**すべて結合**．
- **ニューラルネットワーク**は，**結合の重みとバイアスを調整**して，**望みの出力を得る**．

ニューラルネットワークの**学習**では，**データを利用**して，**望みの出力が得られるよう**に，**結合の重みとバイアスの調整**が行われる．画像認識や自然言語処理などに広く利用されている．

10-3. 畳み込みの仕組み

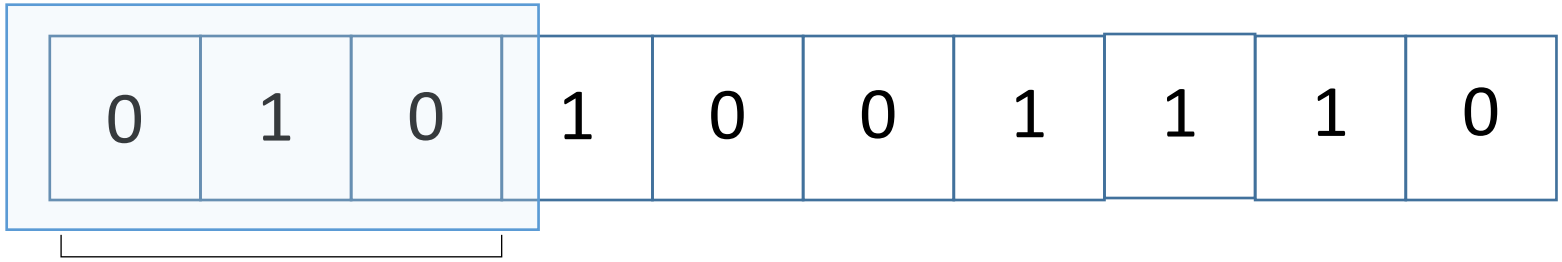
畳み込み

畳み込みは，あるデータを移動しながら，**カーネル**と重ね合わせる．重ね合わせの結果は１つの値になる．

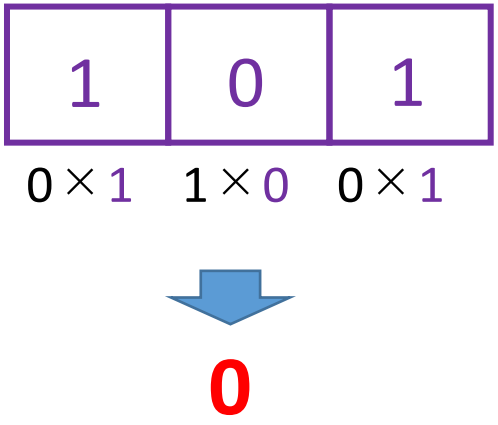


畳み込みの例

データ

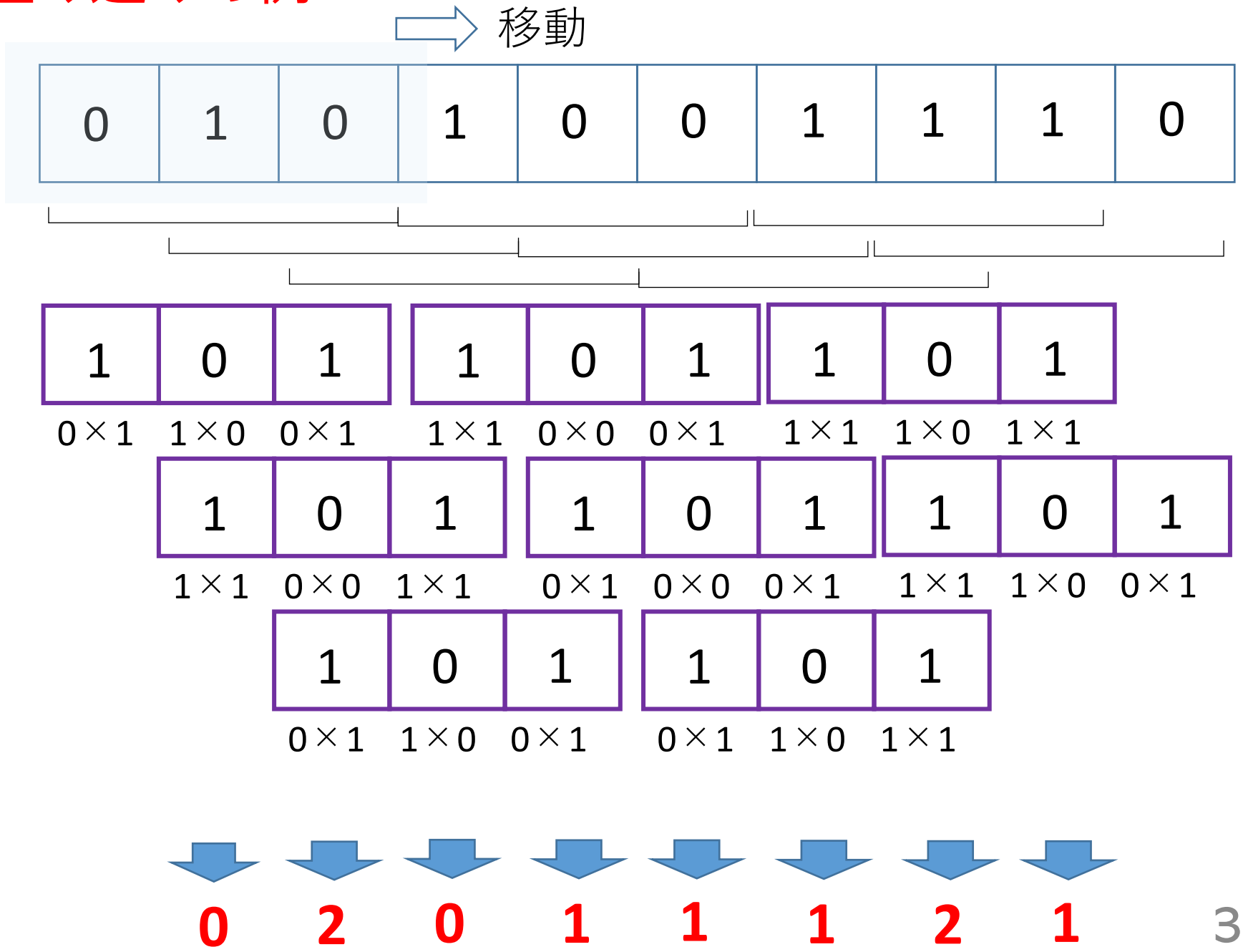


カーネル

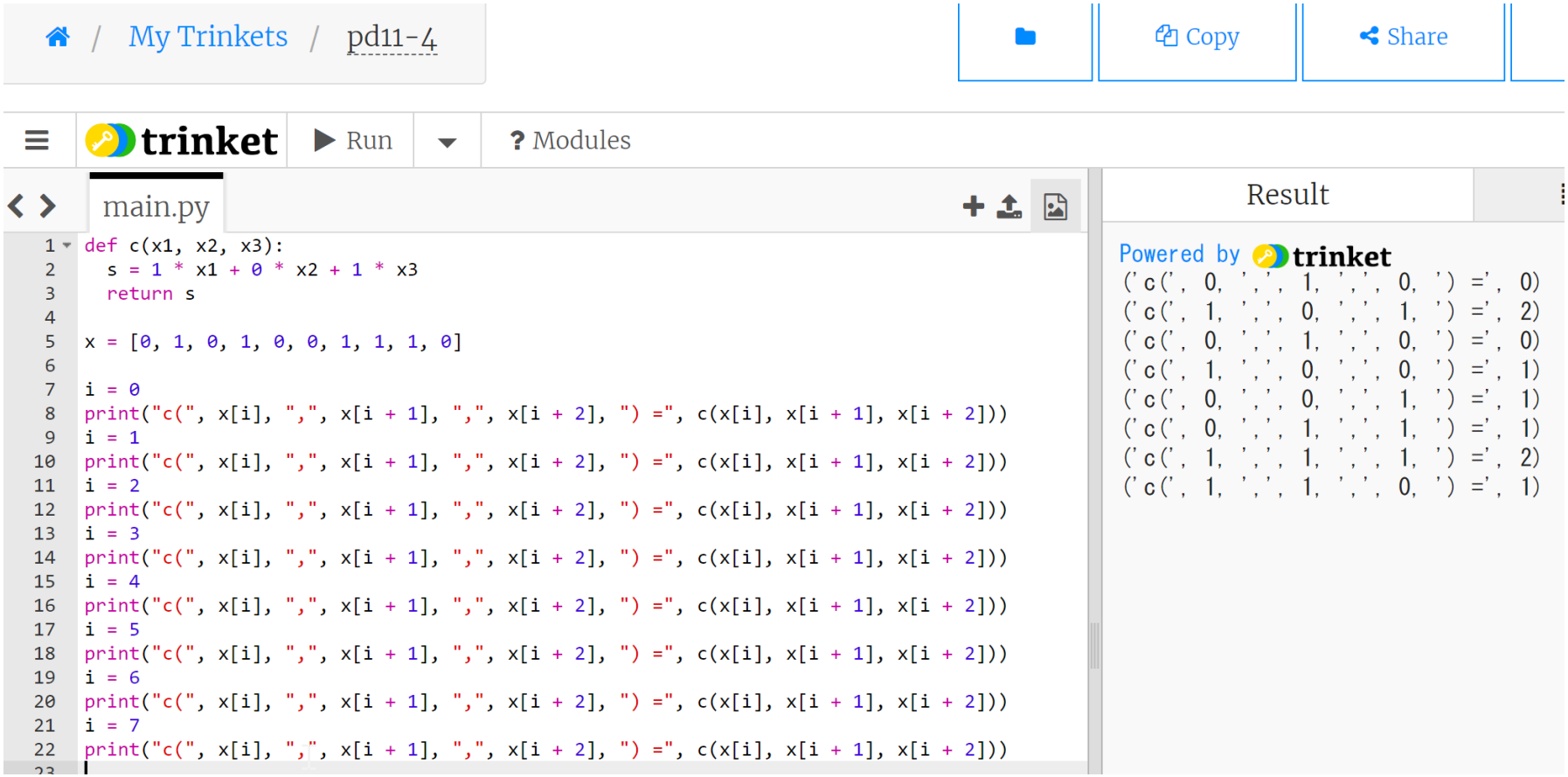


重ね合わせの結果：　0×1 + 1×0 + 0×1 = 0

畳み込みの例


移動
0 1 0 1 0 0 1 1 1 0
1 0 1
0×1 1×0 0×1
1 0 1
1×1 0×0 0×1
1 0 1
1×1 0×0 1×1
1 0 1
0×1 0×0 0×1
1 0 1
1×1 1×0 0×1
1 0 1
0×1 1×0 0×1
1 0 1
0×1 1×0 1×1
0 2 0 1 1 1 2 1

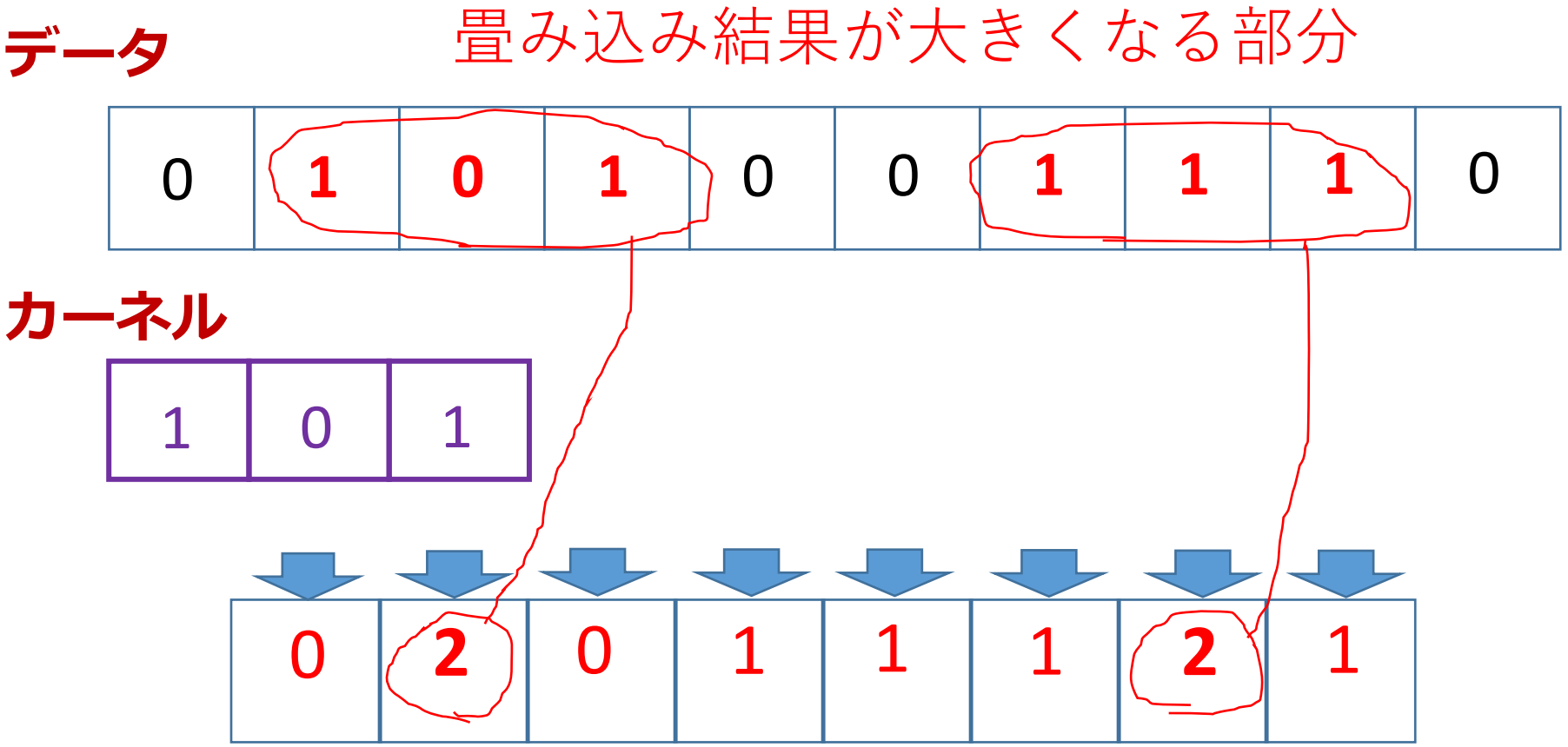
URL: https://trinket.io/python/3419fd07e4



1, 0, 1 や 1, 1, 1 に強く反応することを確認してください

畳み込み

畳み込みは，「**特定のパターンに強く反応する**」と考えることもできる



畳み込みの用途

人工知能の他，画像，音声，その他信号の処理などに広く応用されている

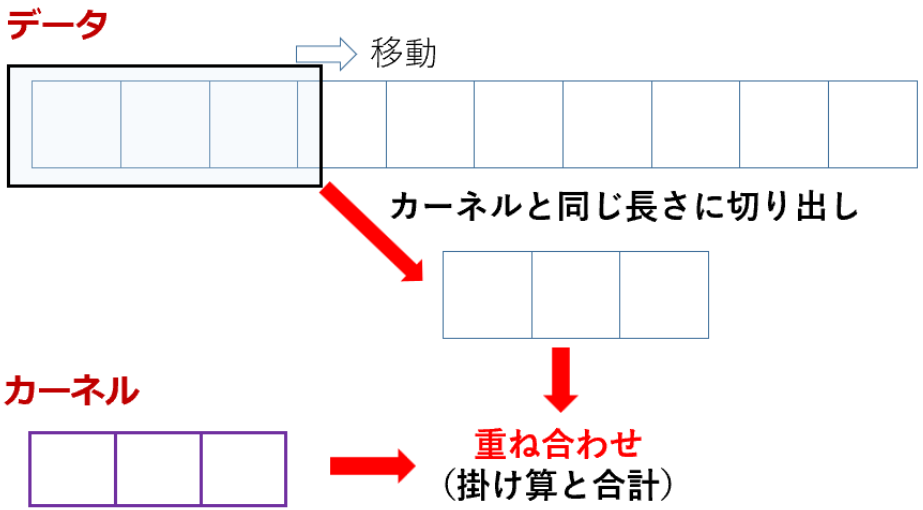
- パターンの分析
  データの中からパターンを発見

- 周波数での分析，処理
  特定の周波数のみ抜き出す

など

畳み込みのまとめ

- **畳み込み**は，**あるデータを移動**しながら，**カーネル**と重ね合わせる．



- **カーネル**は

- **重ね合わせ**は，同じ長さの２つのデータについて，要素同士の掛け算の合計．

10-4. 画像での畳み込み

画像の畳み込み

Input

0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1

元画像（5 × 5 マス）

Filter / Kernel

1	0	1
1	1	1
0	0	1

カーネル（3 × 3 マス）

出典: https://serokell.io/blog/introduction-to-convolutional-neural-networks

画像での畳み込み

0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1

元画像（５×５マス）

1	0	1
1	1	1
0	0	1

カーネル
（３×３マス）

0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1

切り出し（３×３マス）

カーネルと同じサイズで切り出す

切り出した部分とカーネルの掛け算の合計

0×1	**1×0**	**1×1**
0×1	**1×1**	**1×1**
0×0	**1×0**	**1×1**

合計: 4（これが畳み込み結果）

畳み込み

画像での畳み込み

0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1

元画像（５×５マス）

1	0	1
1	1	1
0	0	1

カーネル
（３×３マス）

0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1

切り出し（３×３マス）

0×1	1×0	1×1
0×1	1×1	1×1
0×0	1×0	1×1

合計: 4

1×1	1×0	0×1
1×1	1×1	0×1
1×0	1×0	0×1

合計: 3

4	3	

畳み込み結果

切り出し領域を横にずらす

畳み込み結果

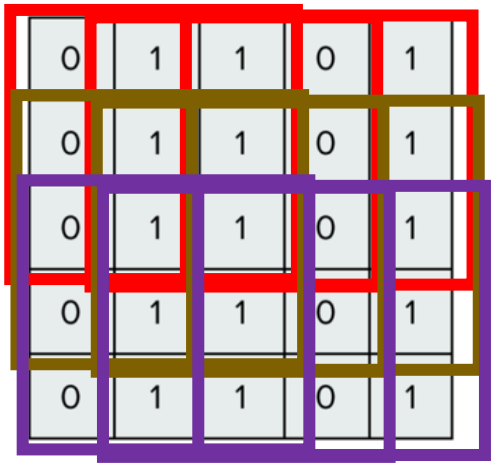
画像での畳み込み

0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	1	1	0	1

元画像（５×５マス）

1	0	1
1	1	1
0	0	1

カーネル
（３×３マス）



切り出し（３×３マス）

画像全体について
畳み込み

4	**3**	**5**
4	**3**	**5**
4	**3**	**5**

畳み込み結果

切り出し領域を縦横にずらす

画像の畳み込みを行う Python プログラムの例

```
x = [[0, 1, 1, 0, 1],
     [0, 1, 1, 0, 1],
     [0, 1, 1, 0, 1],
     [0, 1, 1, 0, 1],
     [0, 1, 1, 0, 1]]
k = [[1, 0, 1],
     [1, 1, 1],
     [0, 0, 1]]
y = [[0, 0, 0],
     [0, 0, 0],
     [0, 0, 0]]

def conv(i, j):
    return x[i + 0][j + 0] * k[0][0] + x[i + 0][j + 1] * k[0][1] + x[i + 0][j + 2] * k[0][2] ¥
    + x[i + 1][j + 0] * k[1][0] + x[i + 1][j + 1] * k[1][1] + x[i + 1][j + 2] * k[1][2] ¥
    + x[i + 2][j + 0] * k[2][0] + x[i + 2][j + 1] * k[2][1] + x[i + 2][j + 2] * k[2][2]

y[0][0] = conv(0, 0)
y[0][1] = conv(0, 1)
y[0][2] = conv(0, 2)
y[1][0] = conv(1, 0)
y[1][1] = conv(1, 1)
y[1][2] = conv(1, 2)
y[2][0] = conv(2, 0)
y[2][1] = conv(2, 1)
y[2][2] = conv(2, 2)

print(y)
```

```
[[4, 3, 5], [4, 3, 5], [4, 3, 5]]
```

https://colab.research.google.com/drive/1pcdD-I5-2VbLizKb3egJIQWXn8tBfOuK?usp=sharing

画像の畳み込みの応用例

- 人工知能以外でも，ぼかし，エッジ抽出などさまざまな処理で，**畳み込み**を使用できる

畳み込みによる
ぼかし

畳み込みによる
エッジ抽出

全体まとめ

- **機械学習は、コンピュータがデータを使用して学習することにより知的能力を向上させる技術。**
  - ルール化やプログラム化が難しい作業や直感的な判断や主観的な要素、経験などの人間にとって容易な作業を目標とする。
- **ニューラルネットワークはユニットがつながり、ユニット間の結合から構成されるネットワーク。**
  - ユニットは入力の重みづけ、合計とバイアス、活性化関数の適用を行い、層構造を形成。
  - ニューラルネットワークは結合の重みとバイアスを調整することで望みの出力を得る。
- **畳み込みはデータを移動しながらカーネルと重ね合わせる操作。**
  - 畳み込みでは、カーネルと同じ長さにデータを切り出し、要素同士の掛け算の合計を求めることで重ね合わせを行い。
  - 畳み込みは画像処理など使用され、特徴抽出やパターン認識などに利用される。

授業の学ぶ意義と満足感

- 機械学習やニューラルネットワークの仕組みを理解。AIの可能性に対する興味と期待の高まり。
- ニューラルネットワークの構成要素や学習原理を具体的に学び、AIの設計・実装に必要な知識を修得。
- 実際のプログラム、学習データを確認し、AIの動作を直感的に把握。アクティブラーニングの楽しさを実感。
- 機械学習の最先端に触れる。AI時代に活躍できる人材を目指すモチベーションが高まり、情報工学で社会に貢献したいという志を高める。