ae-5.ニューラルネットワークとディープラーニング入門：基礎理論から実践まで

（AI 演習）（全１５回）

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

金子邦彦

人工知能（AI）の学習のための指針

1. **実践重視**： AIツールを実際に使用し、機能に慣れる

2. **エラーを恐れない**： 実行においては、エラーの発生の可能性がある．エラーを恐れず，むしろ学習の一部として捉えるポジティブさが大切．

3. **段階的学習**：基礎から応用へと段階的に学習を進め，AIの可能性を前向きに捉える

アウトライン

1. イントロダクション（ニューロンと脳の構造）
2. ニューラルネットワークの基本構造
3. 活性化関数とその役割
4. フォワードプロパゲーション
5. バックプロパゲーション
6. ディープラーニングの特徴と応用

trinket

- **Trinket** は**オンライン**の **Python、HTML** 等の**学習サイト**
- ブラウザで動作
- 有料の機能と無料の機能がある
- **自分が作成した Python プログラムを公開し、他の人に実行してもらうことが可能**（そのとき、書き替えて実行も可能）

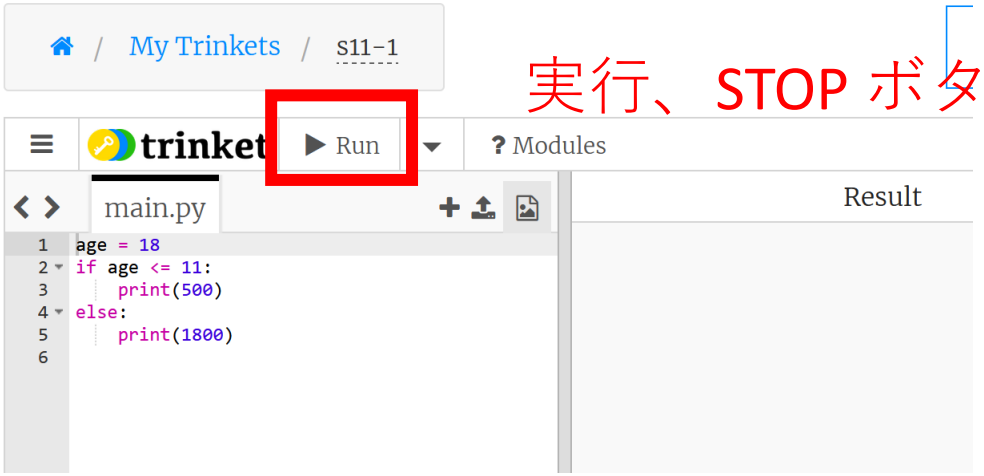
- **Python の標準機能**を登載、その他、次の外部ライブラリがインストール済み

matplotlib.pyplot, numpy, processing, pygal

trinket でのプログラム実行

- **trinket は Python, HTML などのプログラムを書き実行できる**サイト
- https://trinket.io/python/0fd59392c8

のように、違うプログラムには違う URL が割り当てられる

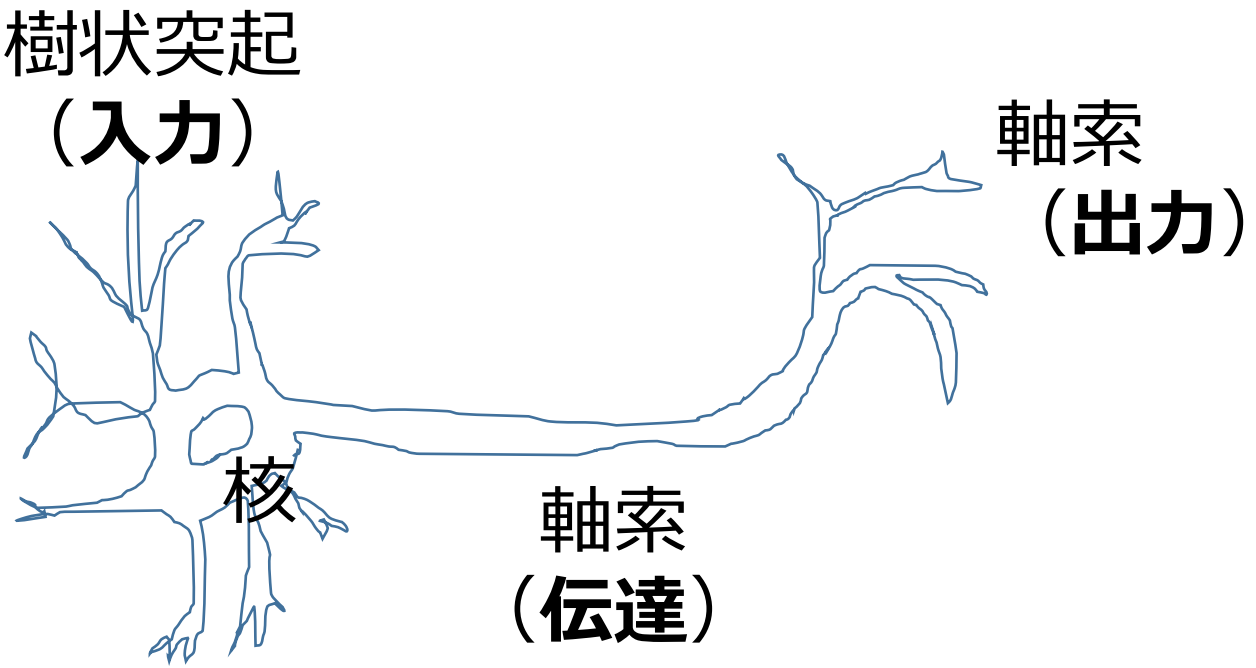


- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

5-1. イントロダクション（ニューロンと脳の構造）

ニューロン

- **ニューロン**は，**脳や神経系**において，**情報伝達，記憶，情報処理**を行う基本要素.
- ニューロン**は樹状突起で入力を受け取り，軸索で信号を伝達**する．核を中心に，入力部（樹状突起），情報処理部（細胞体），出力部（軸索）から構成される



脳のニューロン

脳のニューロンの数（推定値）

- カタツムリ　11,000
- ロブスター　100,000
- アリ　250,000
- カエル　16,000,000
- ハツカネズミ 71,000,000
- タコ　500,000,000
- ネコ　760,000,000
- **ヒト　86,000,000,000**
  ～ 100,000,000,000
- アフリカゾウ 257,000,000,000

Wikipeida の記事:
https://ja.wikipedia.org/wiki/%E5%8B%95%E7%89%A9%E3%81%AE%E3%83%8B%E3%83%A5%E3%83%BC%E3%83%AD%E3%83%B3%E3%81%AE%E6%95%B0%E3%81%AE%E4%B8%80%E8%A6%A7 より

- 生物の脳には，種によって異なる数のニューロンが存在
- 人間の脳には**約860億～1000億個**のニューロンが存在
- ニューロンが複雑なネットワークを形成している

脳や神経系の研究の進展

1900年頃: ニューロンによる情報伝達の理解.

1980年頃: **脳機能マッピングによる脳の機能解明（どの部分がどの機能を持つか）**．生きた脳を対象とした測定が行われるようになってきた.

2010年頃：**異なる脳領域の連携**の観察

脳の仕組みと機能について**洞察**が得られるように

ニューラルネットワーク

ニューラルネットワークは，**入力の重みづけ，合計とバイアス，活性化関数の適用**を行う**ニューロン**が**ネットワーク**を形成する

各ニューロンは複数の入力を受け取り，重み付け，合計とバイアスを適用した後，活性化関数によって出力を決定

ニューラルネットワークの歴史

1940年代： **ニューラルネットワークの誕生**

1960年代： **バックプロパゲーションの誕生**（ニューラルネットワークの**学習の基本的な仕組み**）

1986年：Rumelhartらによるバックプロパゲーションの再発見

2010年代： 多層化，正規化，ReLU，ドロップアウト
など，ニューラルネットワークの**新技術の登場**
（コンピュータで扱えるニューロン数は年々増加）

ニューラルネットワークの進展傾向

年	コンピュータで扱えるニューロン数の規模
2010年	100,000個
2020年	2,000,000個
2030年	50,000,000個
2040年	1,000,000,000個
2050年	20,000,000,000個

所説あります

2055年ごろには1000憶を超えるかも？

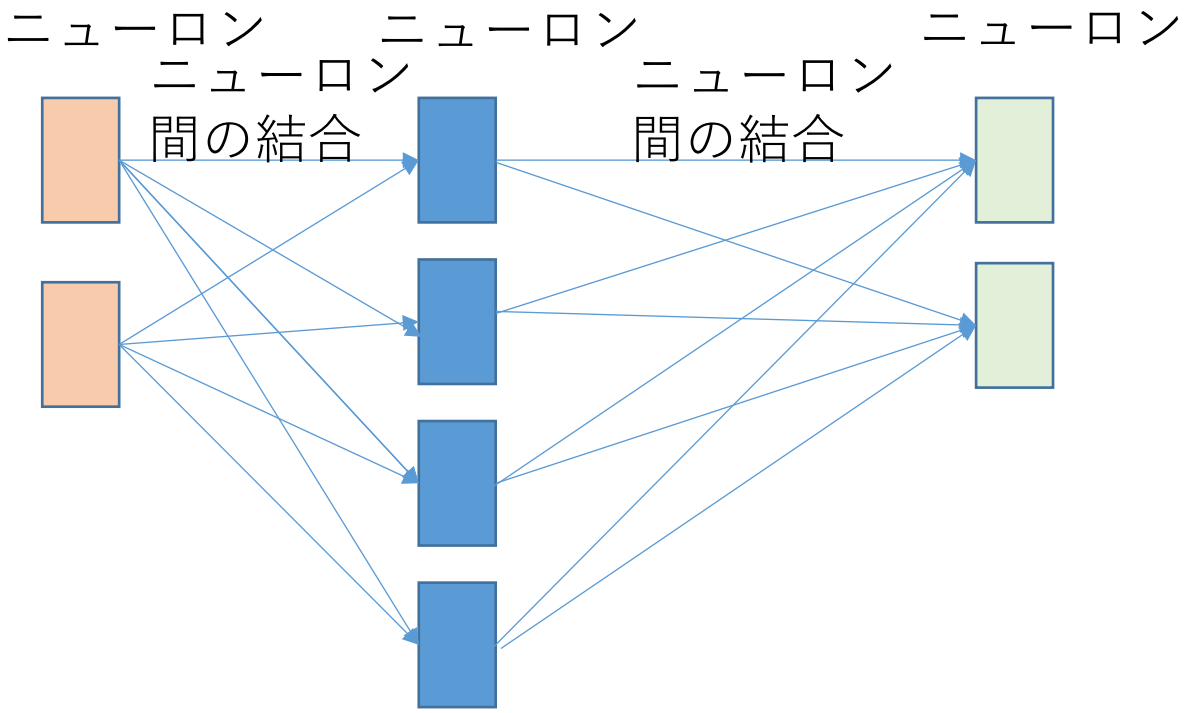
【将来傾向】

- **規模は増加傾向**
- GPU（プロセッサの一種），クラウドコンピューティング（大規模計算の基盤）の**発展**
- 人間の脳の**約860～1000億個**のニューロンを模倣することも、あながち不可能ではないかも

5-2. ニューラルネットワークの基本構造

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**を行う**ニューロン**が**ネットワーク**を形成する



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**: **各入力に対応する重み**を掛ける．例：
   0.3×**0.1**　**-0.5**×**0.8**　**0.2**×**-0.5**
2. **合計とバイアス**: **重みづけした値の合計にバイアス**（数値）を加える．例 **0.03　-0.4　-0.1 の合計は -0.47．これに 0.2 を加える**
3. **活性化関数の適用**: 結果に，**活性化関数**を適用し，**活性度を決定**（**活性度が高いほどニューロンが強く反応している**ことを示す→次のニューロンの入力になる）

0.3　× 0.1　⇒ 0.03

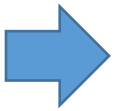
-0.5　× 0.8　⇒ -0.4

0.2　× -0.5　⇒ -0.1

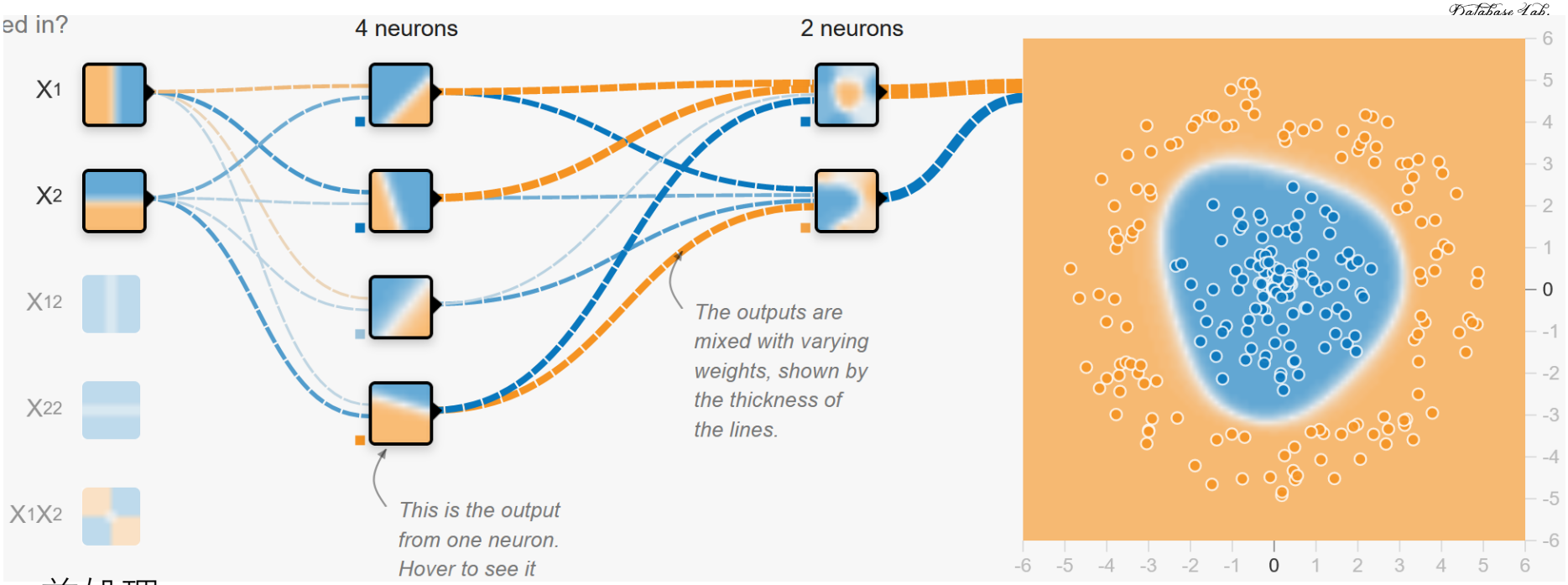
入力　重み

合計
-0.47

バイアス 0.2



-0.27 に
活性化関数
を適用



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

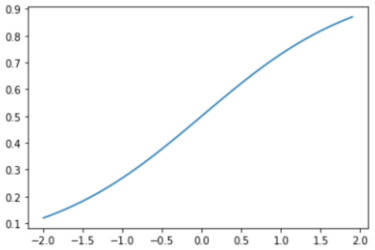
ニューラルネットワーク

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

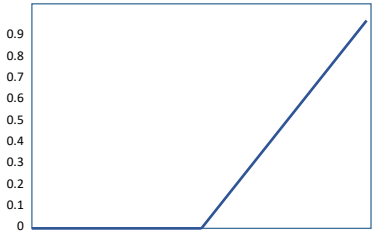
5-3. 活性化関数とその役割

活性化関数

- **ニューロン**の**活性度を決定**する関数
- **入力の重みづけ**，**合計とバイアス**の**結果**に，**活性化関数を適用**し，ニューロンの活性度を決定
- **ReLU**や**シグモイド**など，さまざまな種類



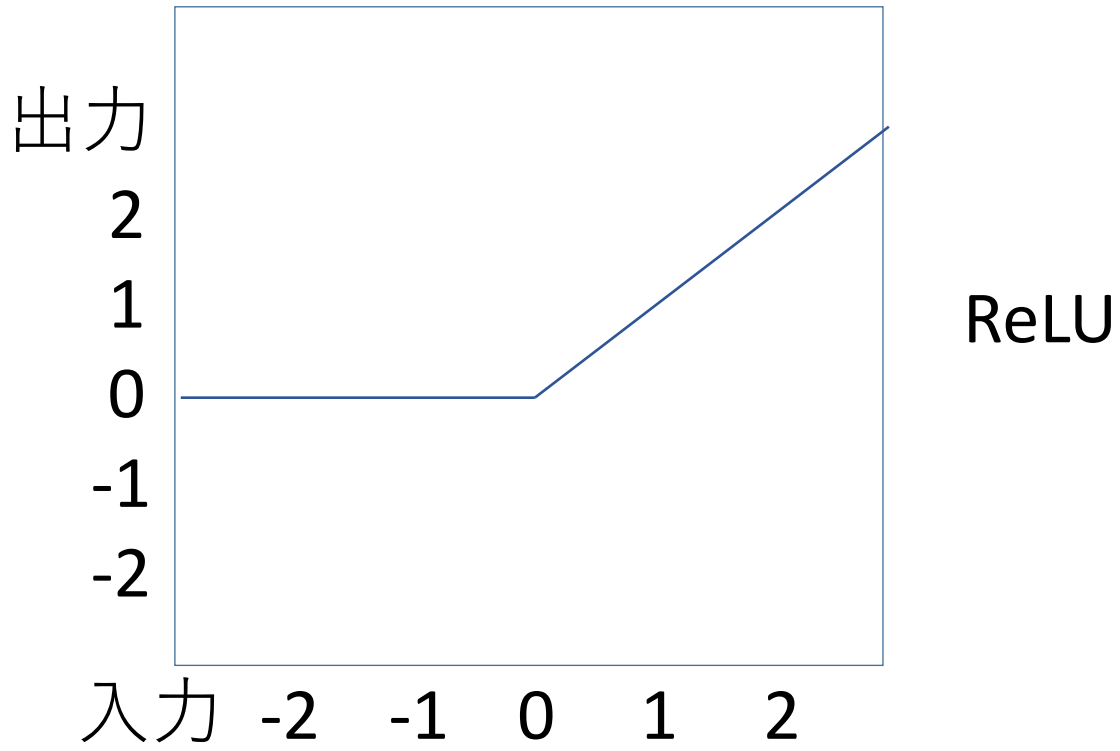
シグモイド



ReLU

ReLU

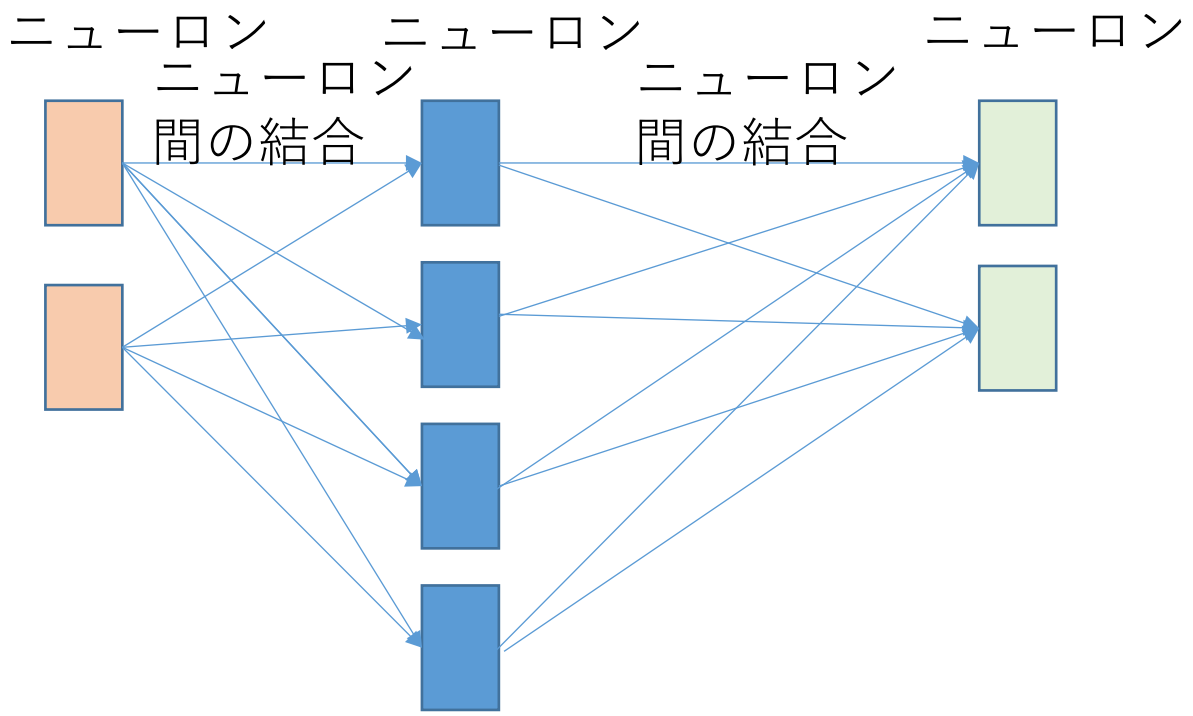
- **2011年に提案**された**活性化関数**
- 入力が**0以下の場合は0を返し，0より大きい場合はその値をそのまま返す**特性を持つ
- シンプルな構造ながら高い性能を示し，現代のディープラーニングで広く使用されている.



5-4. フォワードプロパゲーション

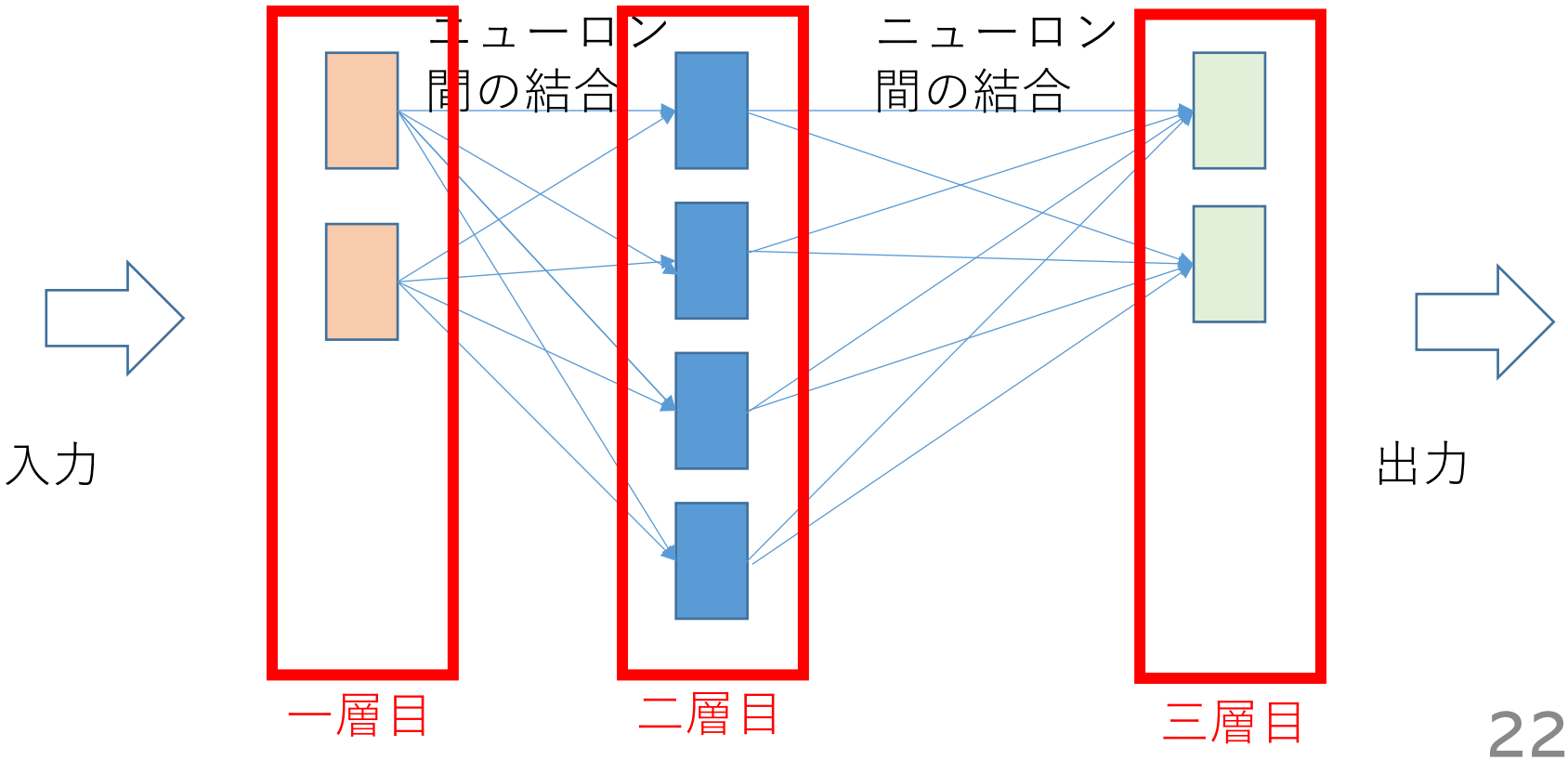
フォワードプロパゲーション

- **フォワードプロパゲーション**は，**データ**が**入力層**から**出力層**へと**順方向に伝播**する**基本的な処理過程**である．
- 各層のニューロンは，**前層からの入力を処理**し，その**結果を次層に伝達**する．



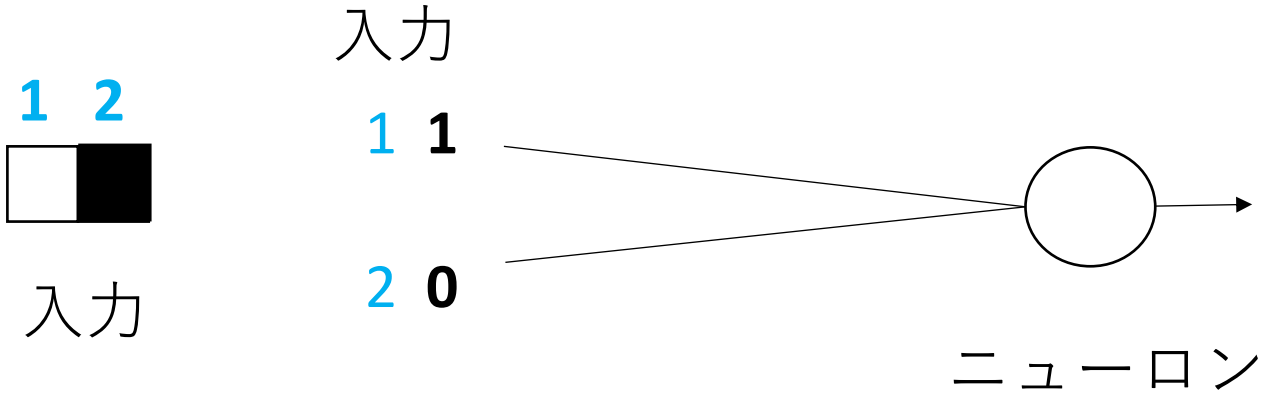
ニューラルネットワークの層構造

ニューラルネットワークの基本構造であり，**複数の層**から成る．**データは入力から出力への一方向**に流れ，**各層**は**同じ種類のニューロンで構成**される．



①入力

数値，複数可　　例：１ ０

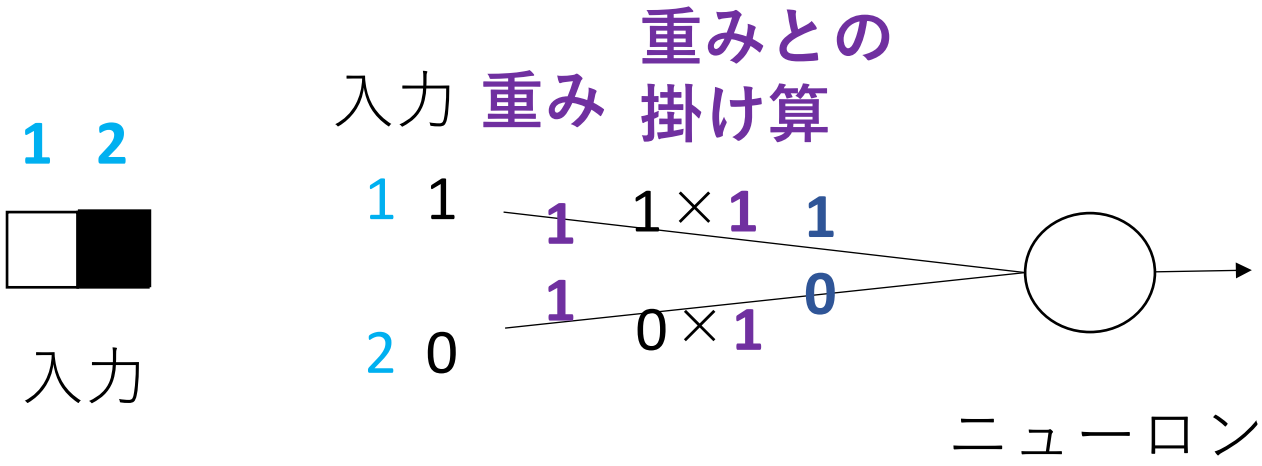


白を１
黒を０とする

②入力の重みづけ

- **ニューロン間の結合**には**重み**が設定され，**信号の伝達強度**を決定する.
- 各入力値に対して，それぞれの対応する**重みを掛ける**

例： **1**×**1**　**0**×**1**

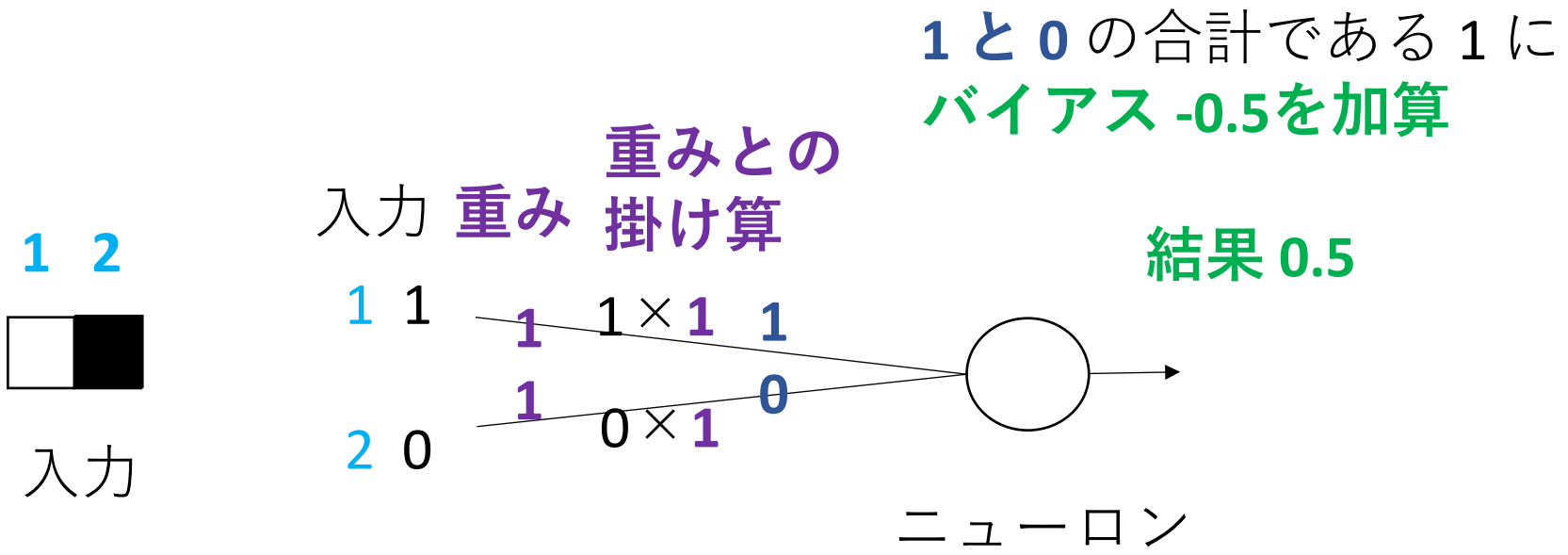


白を 1
黒を 0 とする

③バイアス

- **バイアスはニューロンの活性化のしやすさを調整する.**
- **重みづけした値の合計にバイアス**（数値）を加える.

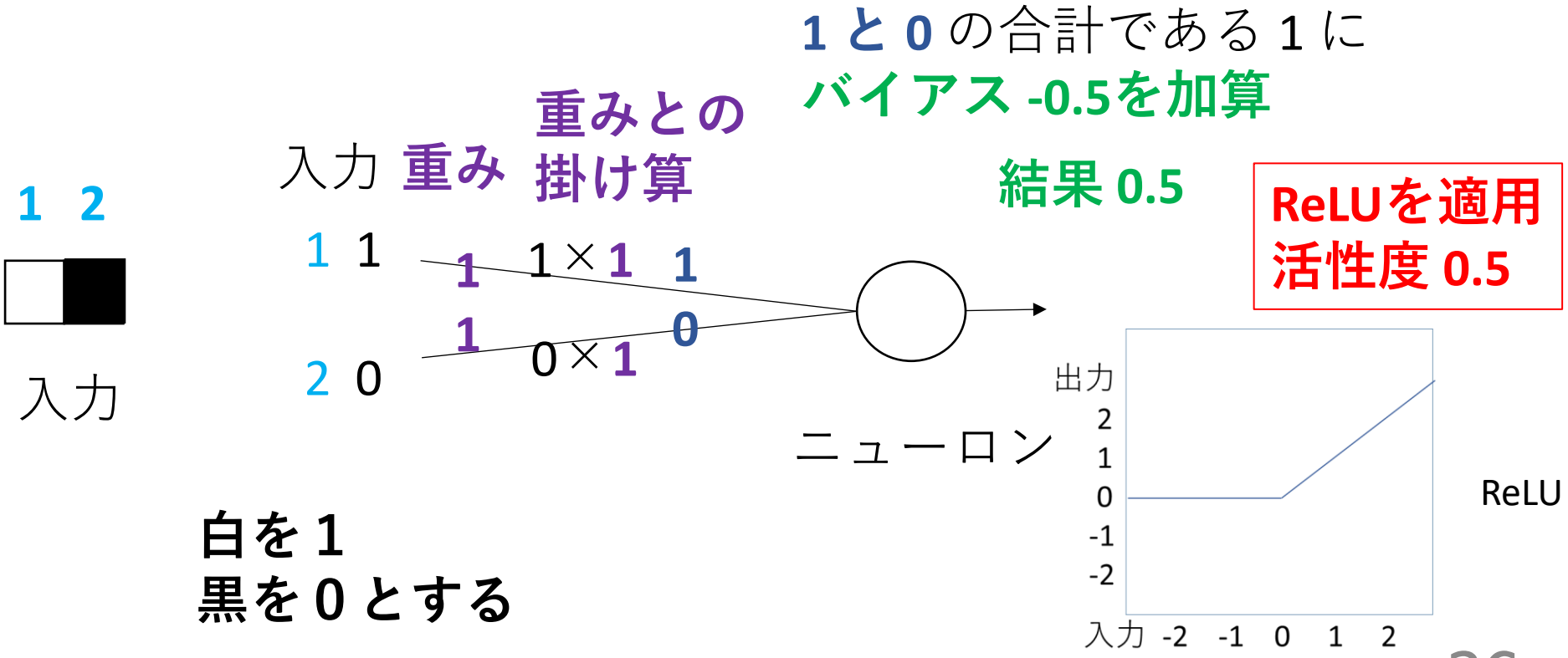
  例 **1 0 の合計は 1. これに -0.5 を加える**



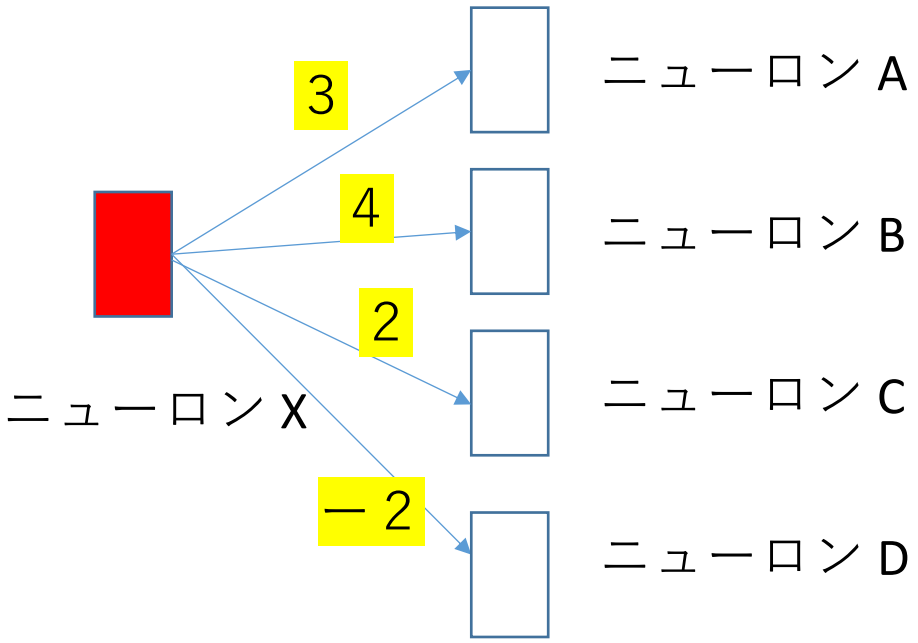
白を 1
黒を 0 とする

④活性化関数の適用

- 結果に，**活性化関数**を適用．**ニューロンの活性度**を得る
  （→次のニューロンの入力になる）
- **値が大きいほどニューロンが活性化している**ことを示す



活性度（数値）は，次のニューロンの入力になる



次の二ューロンの**入力**
＝　**前**の二ューロンの**出力**
×　**結合の重み**

結合の重みは，
結合ごとに**違う値**

演習１　ReLU

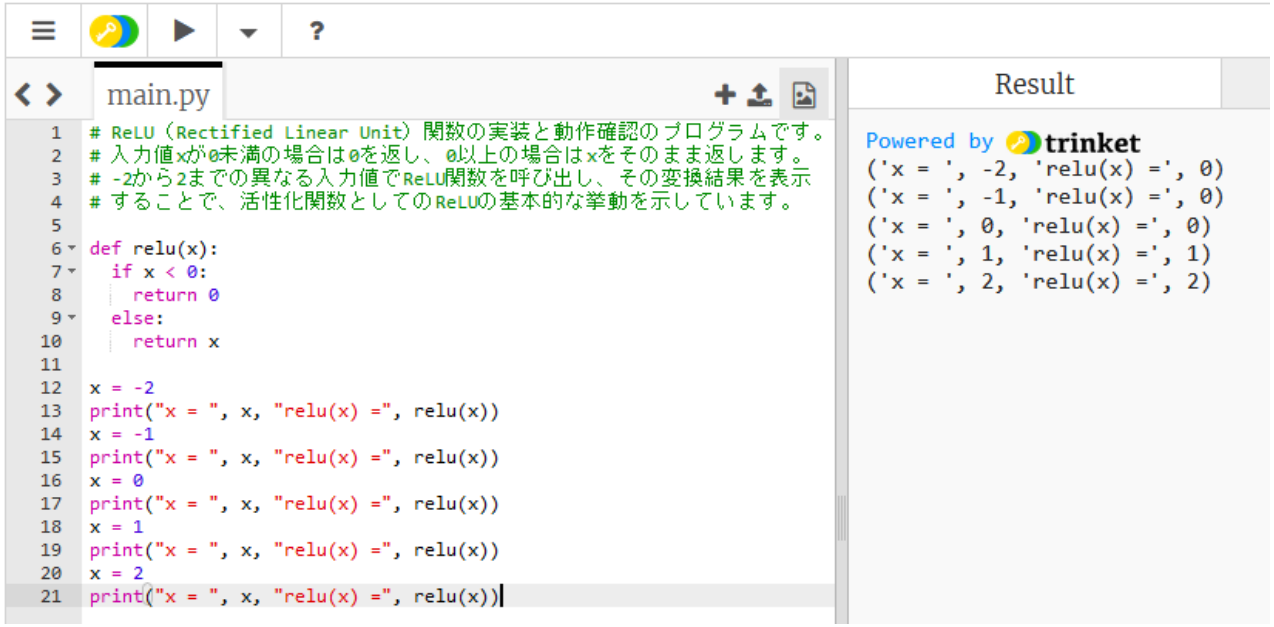
演習の狙い

- **ReLU関数の特徴**（負の入力を0に変換、正の入力はそのまま出力）を**直感的に理解**
- **異なる入力値（負数、0、正数）**で実験して**結果を確認**する体験
- if と else を使うプログラミング（条件分岐）

① trinket の次のページを開く

https://trinket.io/python/61c6503fcada

② 実行結果が，次のように表示されることを確認



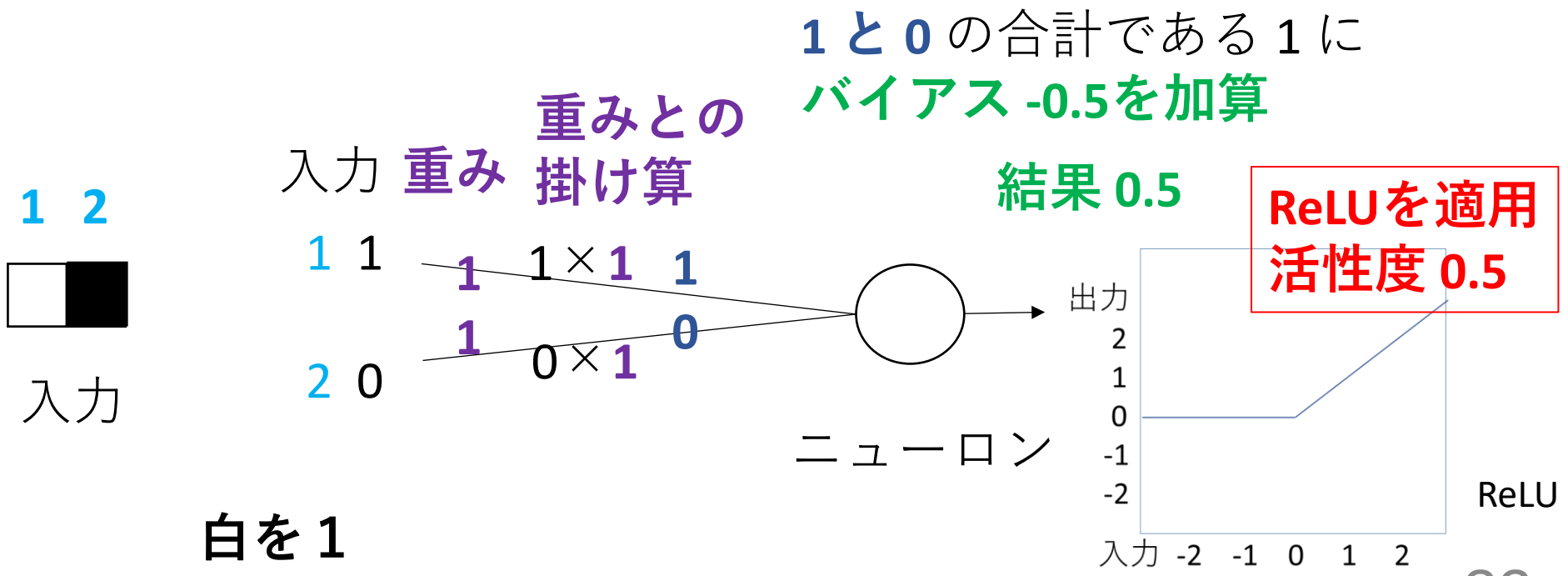
- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

演習２　入力の重みづけ，合計とバイアス，活性化関数の適用

演習の狙い

- **重み，バイアス，活性化関数（ReLU）**という3つの基本要素の役割と連携を具体的に理解できる.
- これは複雑なニューラルネットワークを理解するための重要な第一歩となる.

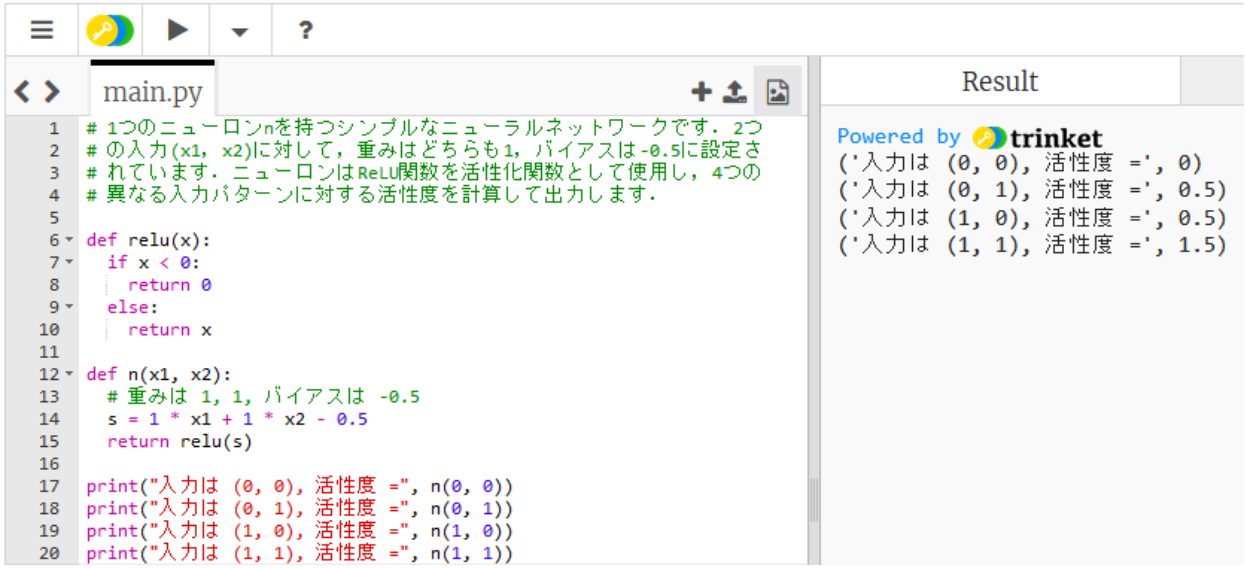
【入力が１０のとき】



① trinket の次のページを開く

https://trinket.io/python/1c339b660ee1

② 実行結果が，次のように表示されることを確認



- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

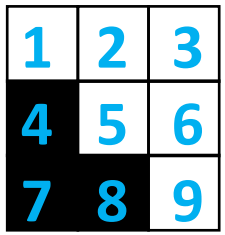
ニューラルネットワークの基本まとめ

- 各**ニューロン**は、前の層からの**入力**を受け取る
- **入力**には**重み**が関連付けられており、**各データがニューロンの活性化に与える影響**を調整する
- **ニューロン**は**バイアス**を持ち、**入力**に**重み**をかけた**総和**に**バイアス**を**加算**する
- **活性化関数**による**処理**を行い、その結果を**次の層のニューロン**に伝える
- ReLUなどの**活性化関数**がある。
- **ニューロン**は**特定のパターン**の**入力**に対して**活性化**し、その**活性度**は**入力**、**重み**、**バイアス**、**活性化関数**によって決まる

ニューラルネットワークによるパターン認識の例

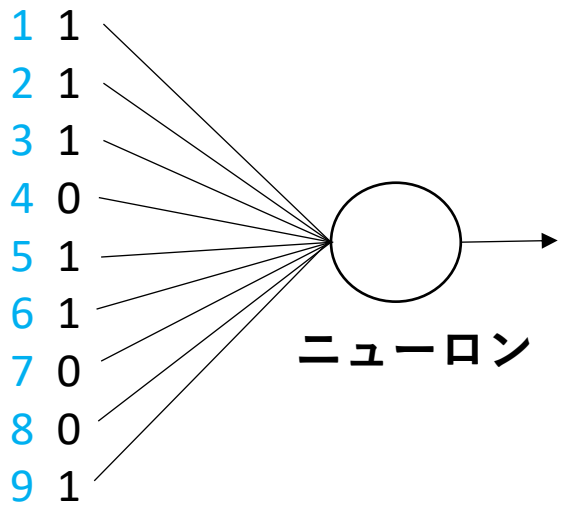
1. **入力の重みづけ**:
   1 × 1, 1 × 1, 1 × 1, 0 × 0, 1 × 1, 1 × 1, 0 × 0, 0 × 0, 1 × 1
2. **合計とバイアス**:
   合計 **6** にバイアス **-5** を足す．結果 **1**
3. **活性化関数の適用**

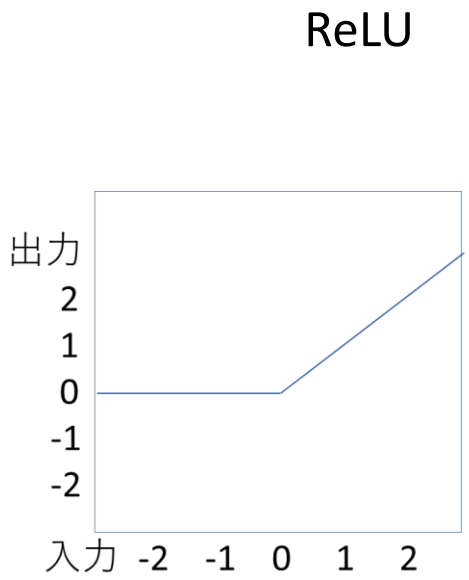
   結果 **1** に **ReLU** を適用した場合は，活性度 **1**



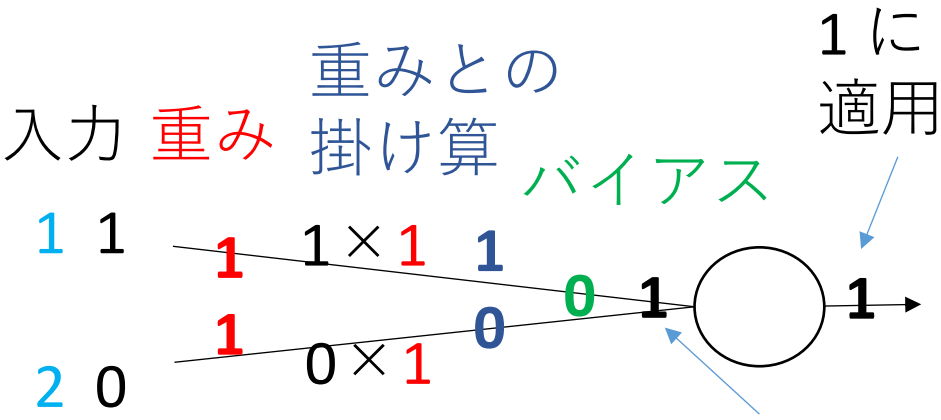
入力

白黒の画像
（画素は 0 または 1）

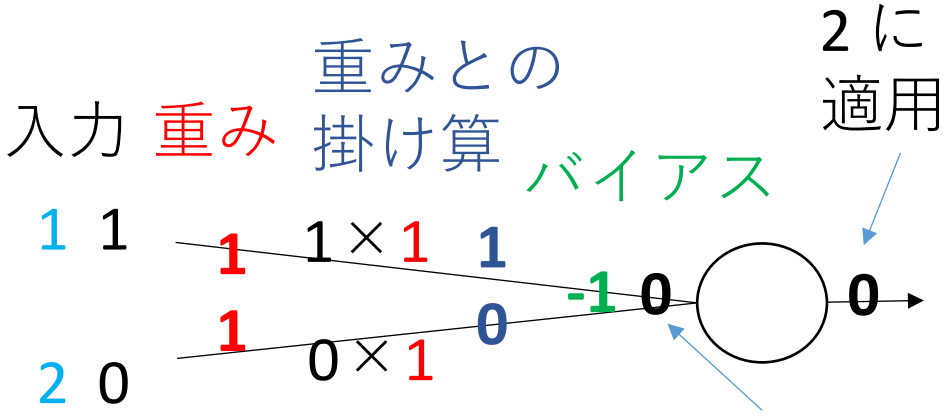




バラメータ（バイアス）の変化による出力の変化

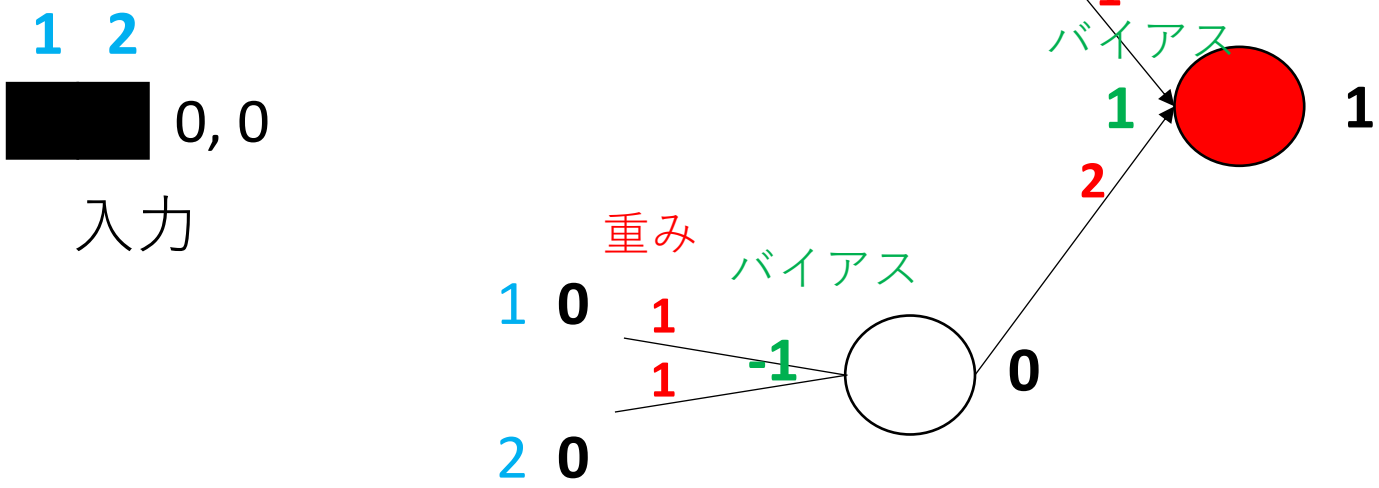


重み 1, 1　バイアス 0

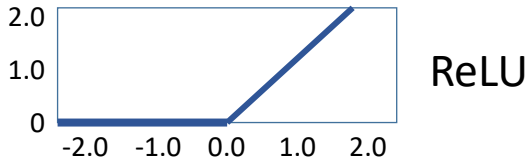


重み 1, 1　**バイアス -1**

大きなプラスの値を
出力している
ニューロン＝活性化



3つのニューロンの活性化関数はすべて ReLU

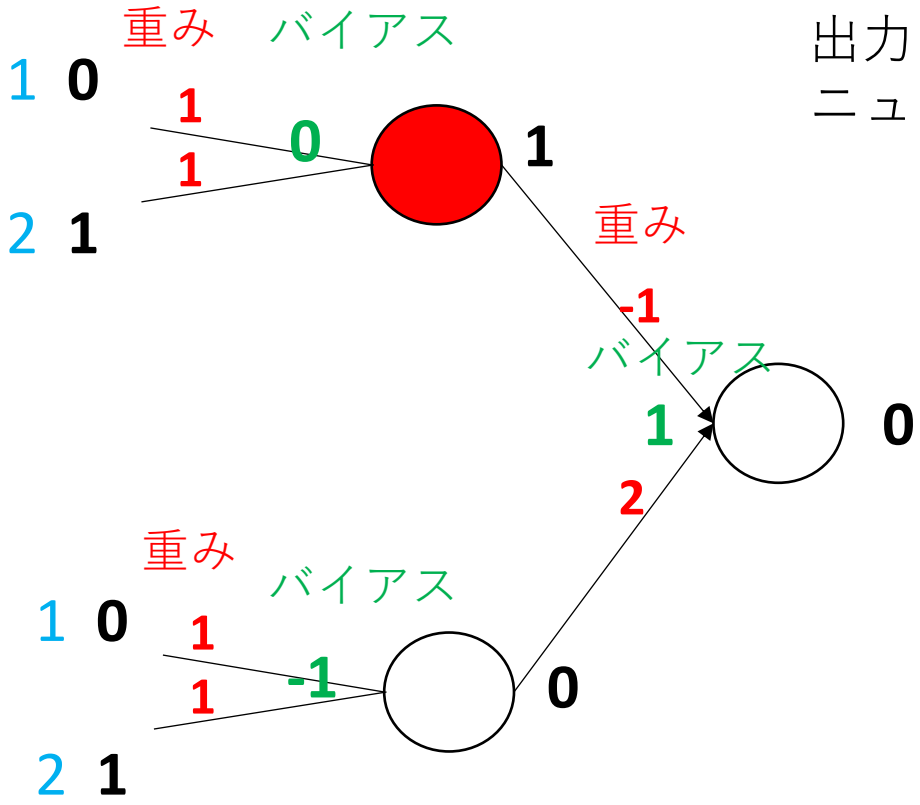


大きなプラスの値を
出力している
ニューロン＝活性化

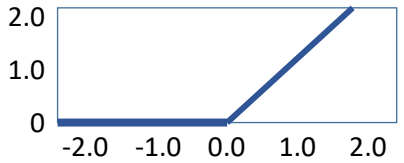
1 2

0, 1

入力

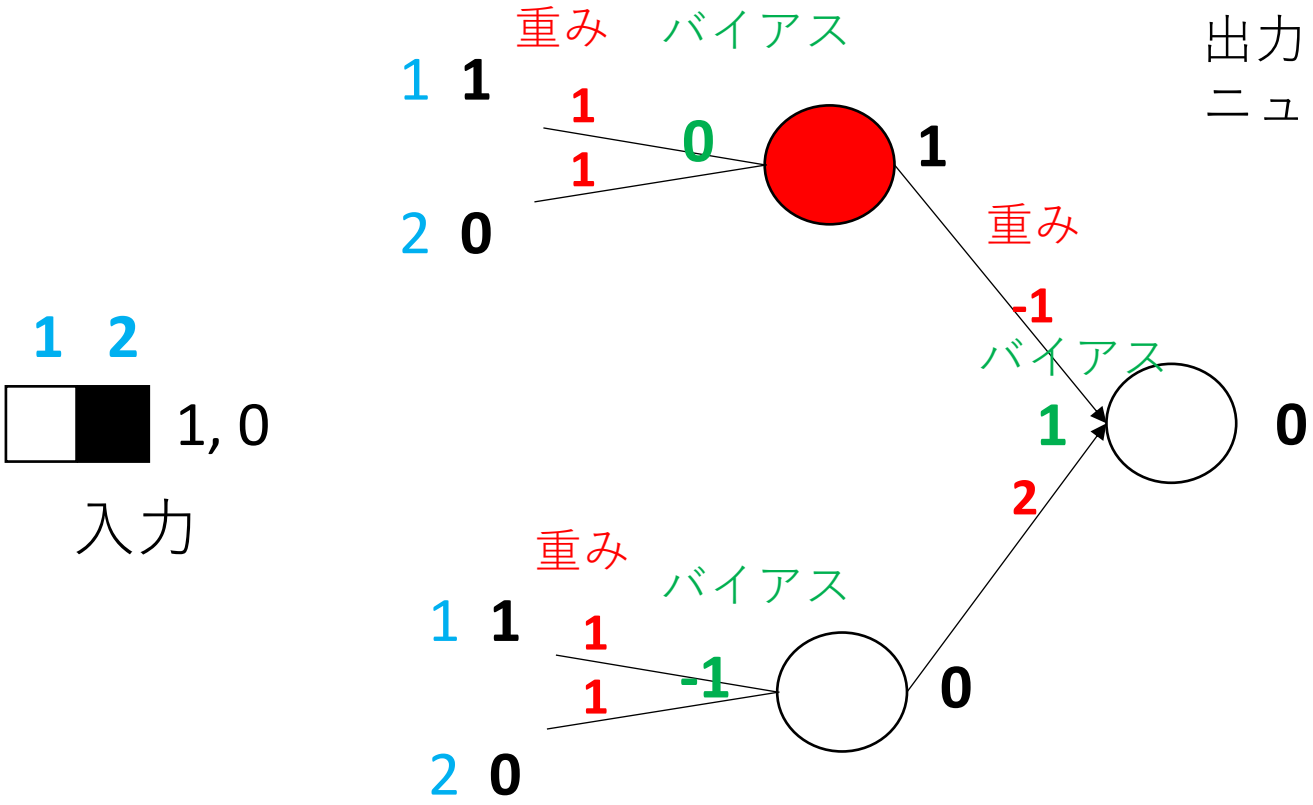


3つのニューロンの活性化関数はすべて ReLU

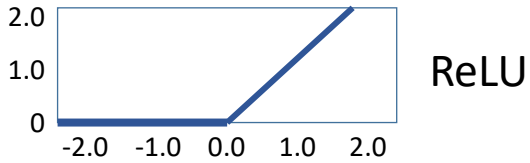


ReLU

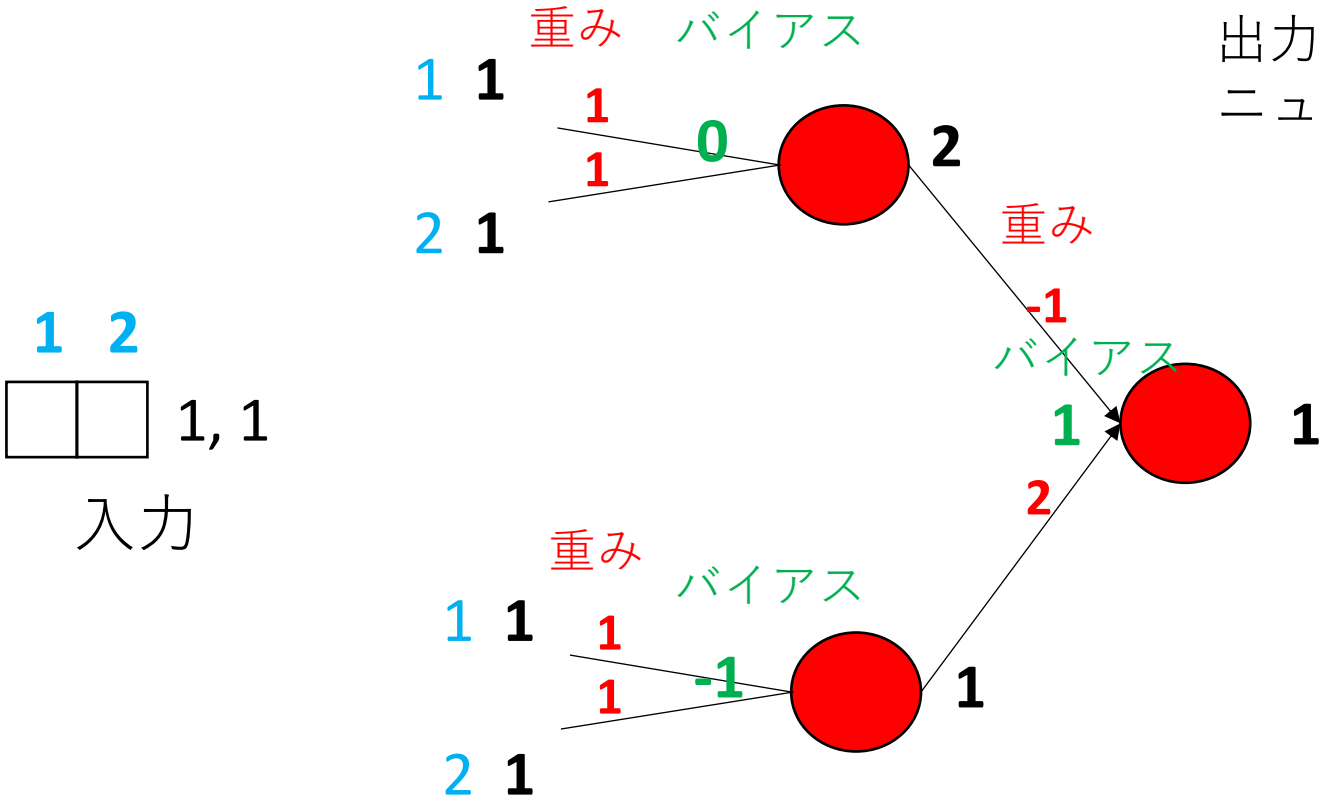
大きなプラスの値を
出力している
ニューロン＝活性化



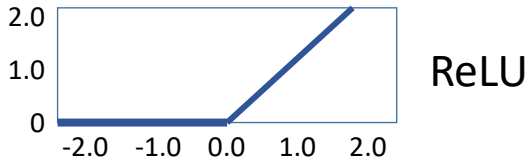
3つのニューロンの活性化関数はすべて ReLU



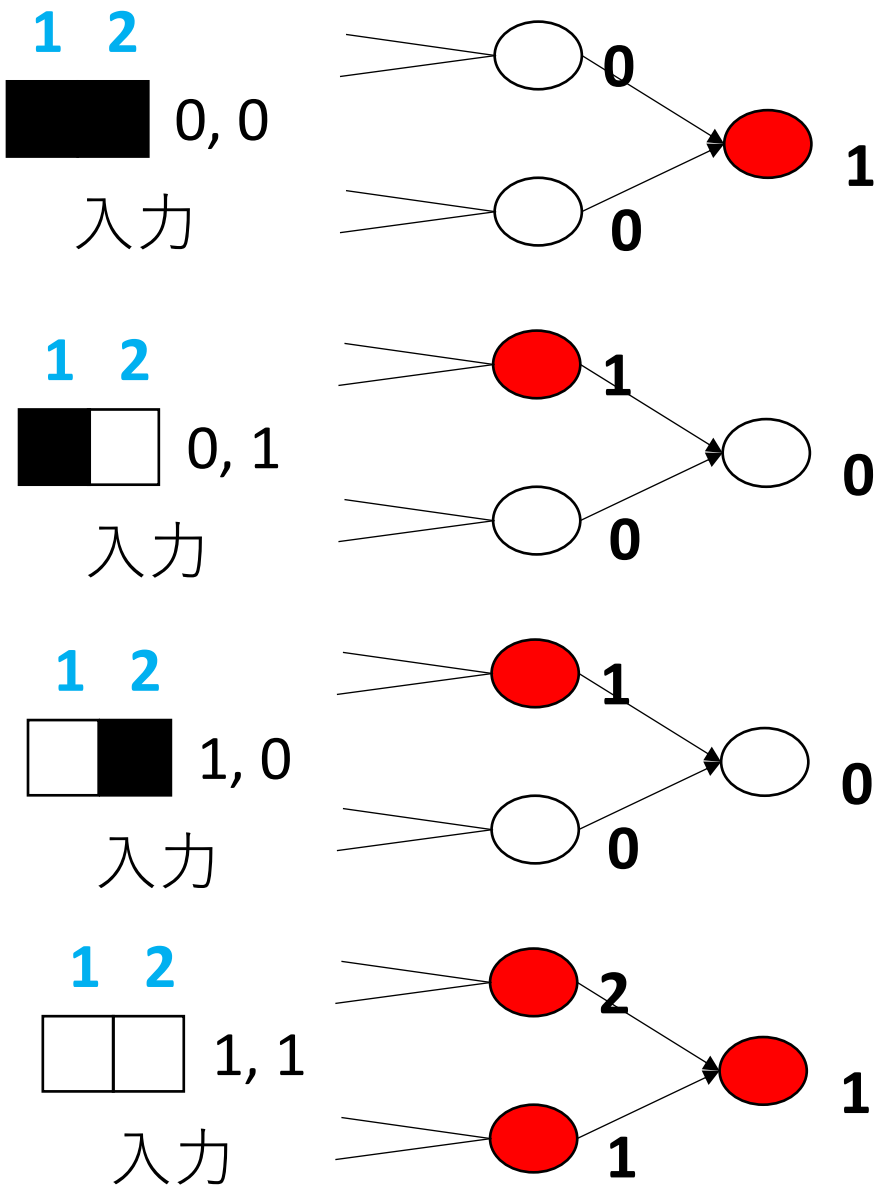
大きなプラスの値を
出力している
ニューロン＝活性化



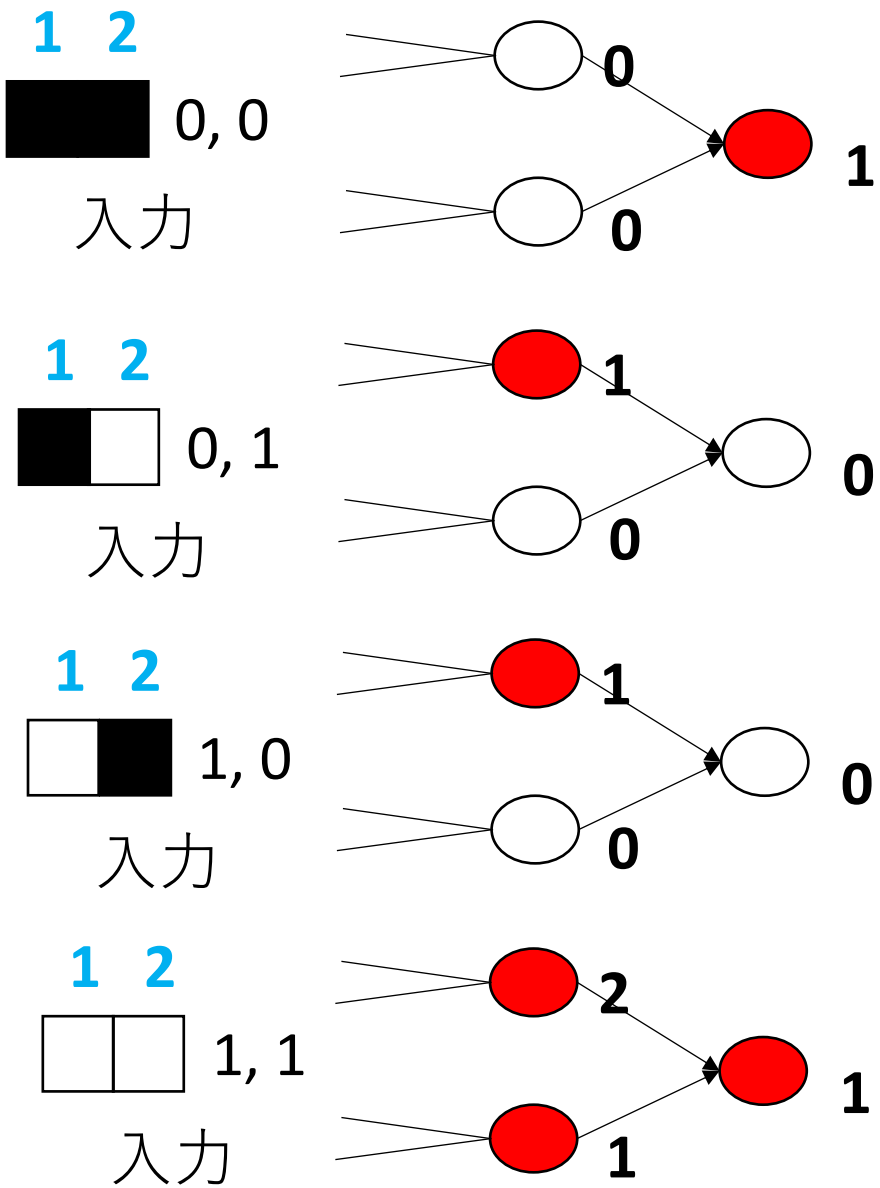
3つのニューロンの活性化関数はすべて ReLU



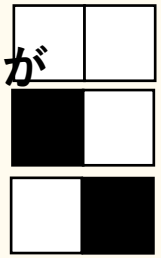
入力の変化による活性度の変化

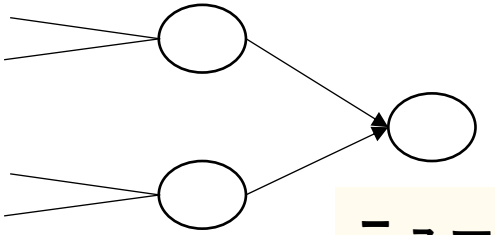


ニューロンは特定のパターンに応じて活性化する

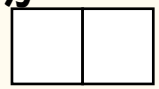


ニューロンが識別するパターン

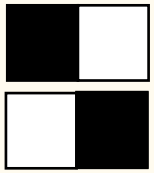




ニューロンが識別するパターン



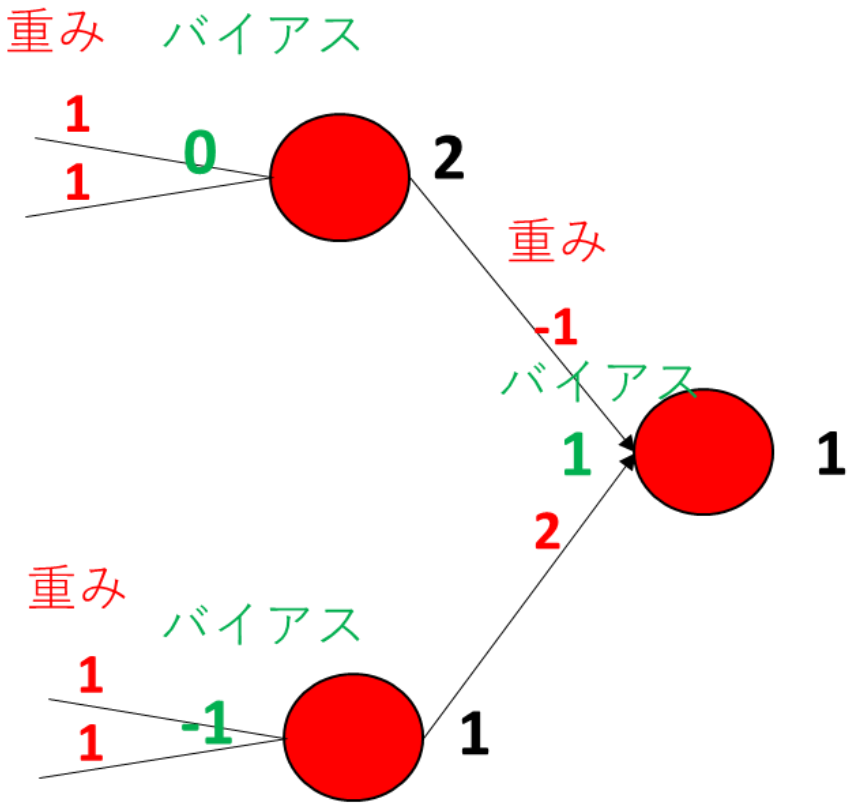
ニューロンが識別するパターン



演習３　ニューラルネットワーク

演習の狙い

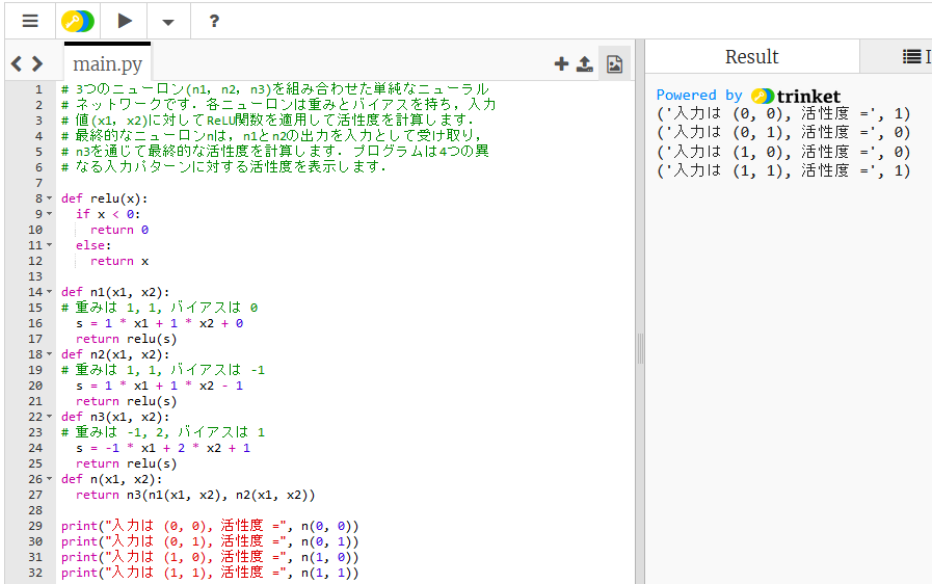
- 複数のニューロンが層を形成
- １層の場合より複雑な判断が可能になる
- ニューラルネットワークによる高度な情報処理の基礎を体験



① trinket の次のページを開く

https://trinket.io/python/3cbc3f3ed057

② 実行結果が，次のように表示されることを確認




```
# 3つのニューロン(n1, n2, n3)を組み合わせた単純なニューラル
# ネットワークです．各ニューロンは重みとバイアスを持ち，入力
# 値(x1, x2)に対してReLU関数を適用して活性度を計算します．
# 最終的なニューロンnは，n1とn2の出力を入力として受け取り，
# n3を通じて最終的な活性度を計算します．プログラムは4つの異
# なる入力パターンに対する活性度を表示します．

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

def n1(x1, x2):
# 重みは 1, 1, バイアスは 0
  s = 1 * x1 + 1 * x2 + 0
  return relu(s)
def n2(x1, x2):
# 重みは 1, 1, バイアスは -1
  s = 1 * x1 + 1 * x2 - 1
  return relu(s)
def n3(x1, x2):
# 重みは -1, 2, バイアスは 1
  s = -1 * x1 + 2 * x2 + 1
  return relu(s)
def n(x1, x2):
  return n3(n1(x1, x2), n2(x1, x2))

print("入力は (0, 0), 活性度 =", n(0, 0))
print("入力は (0, 1), 活性度 =", n(0, 1))
print("入力は (1, 0), 活性度 =", n(1, 0))
print("入力は (1, 1), 活性度 =", n(1, 1))
```

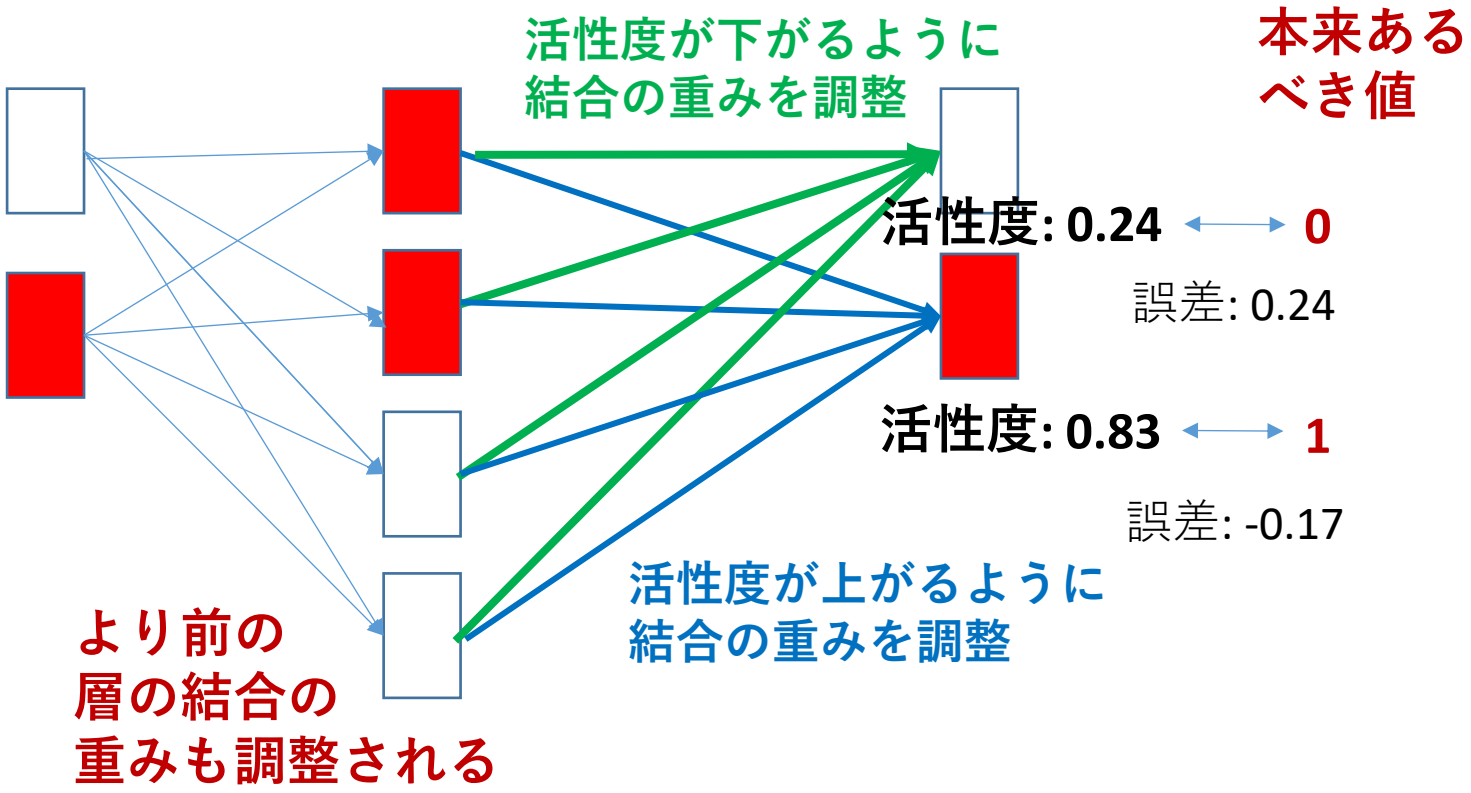
- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

5-5. バックプロパゲーション

バックプロパゲーション

- **バックプロパゲーション**は，**出力の誤差**をもとに**出力層から入力層へ向かって結合の重みとバイアスを調整**する学習の仕組み.
- 誤差を最小化するように**結合の重みとバイアスを自動調整**する
- 1986年にRumelhartらによって効果的な学習手法として確立

ニューラルネットワークの動作イメージ



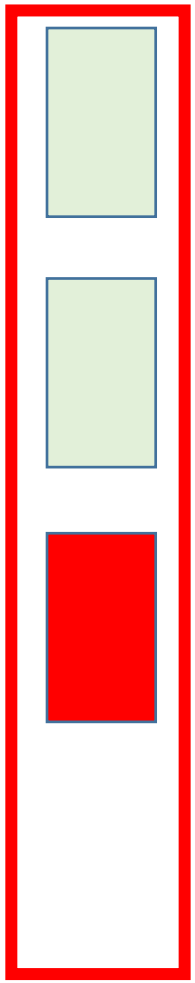
正解と誤差

正解は 2
であるとする

活性度: 0.0068

活性度: 0.1592

活性度: 0.8340



誤差: 0.0068

あるべき値: 0

誤差: 0.1592

あるべき値: 0

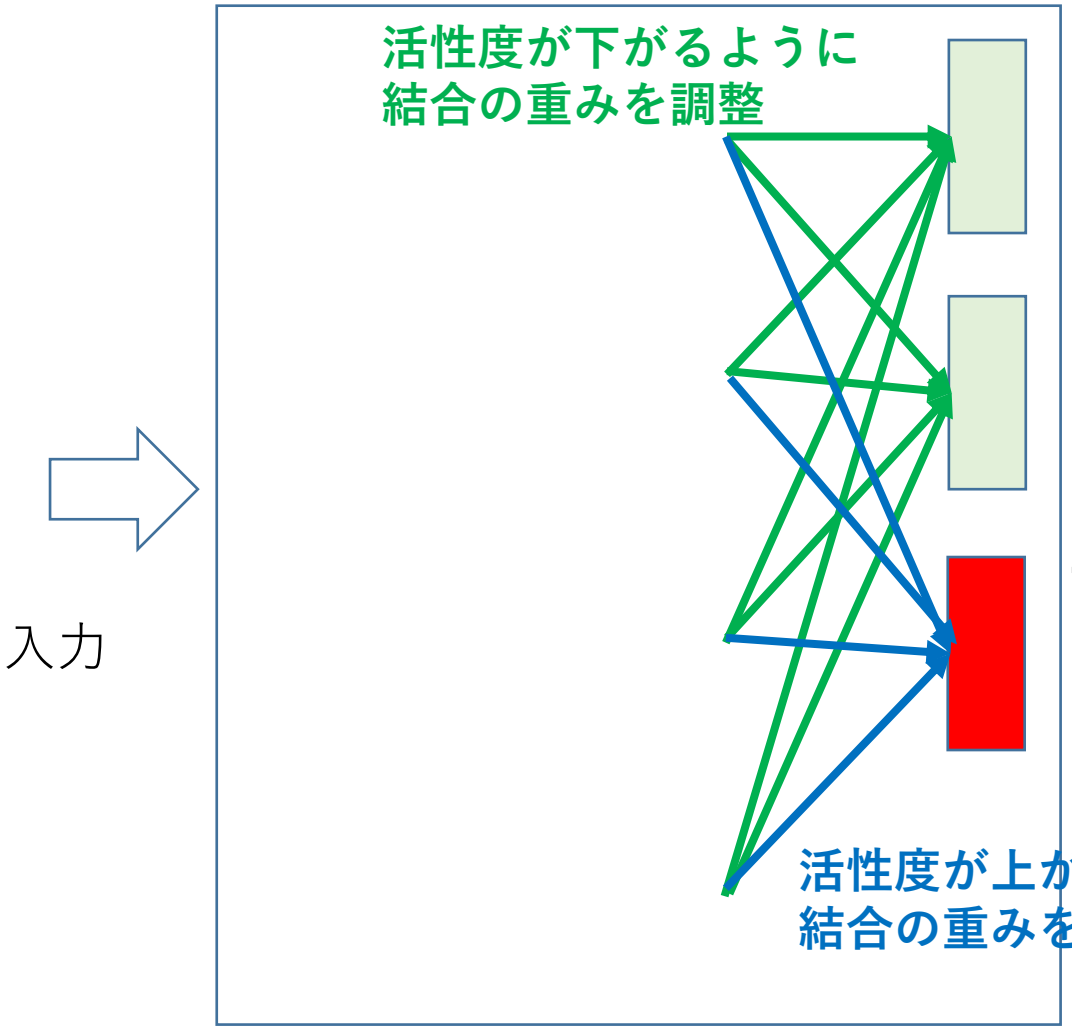
誤差: - 0.1760

あるべき値: 1

誤差をもとに、結合の重みを自動調節

ニューラルネットワークの学習

正解は 2
であるとする

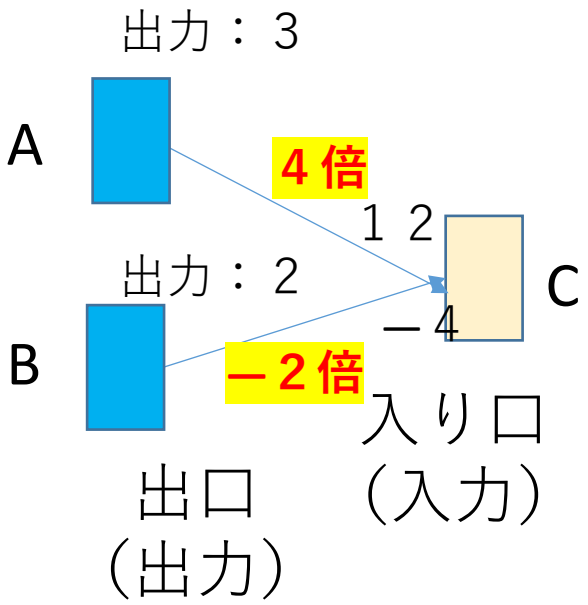


誤差: 0.0068
あるべき値: 0
活性度: 0.0068

誤差: 0.1592
あるべき値: 0
活性度: 0.1592

誤差: - 0.1760
あるべき値: 1
活性度: 0.8340

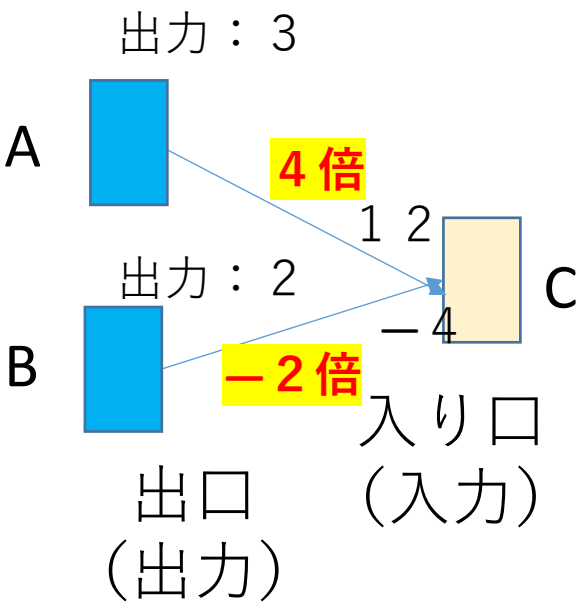
結合の重みの調整



ニューロン間の結合の
　「○○**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

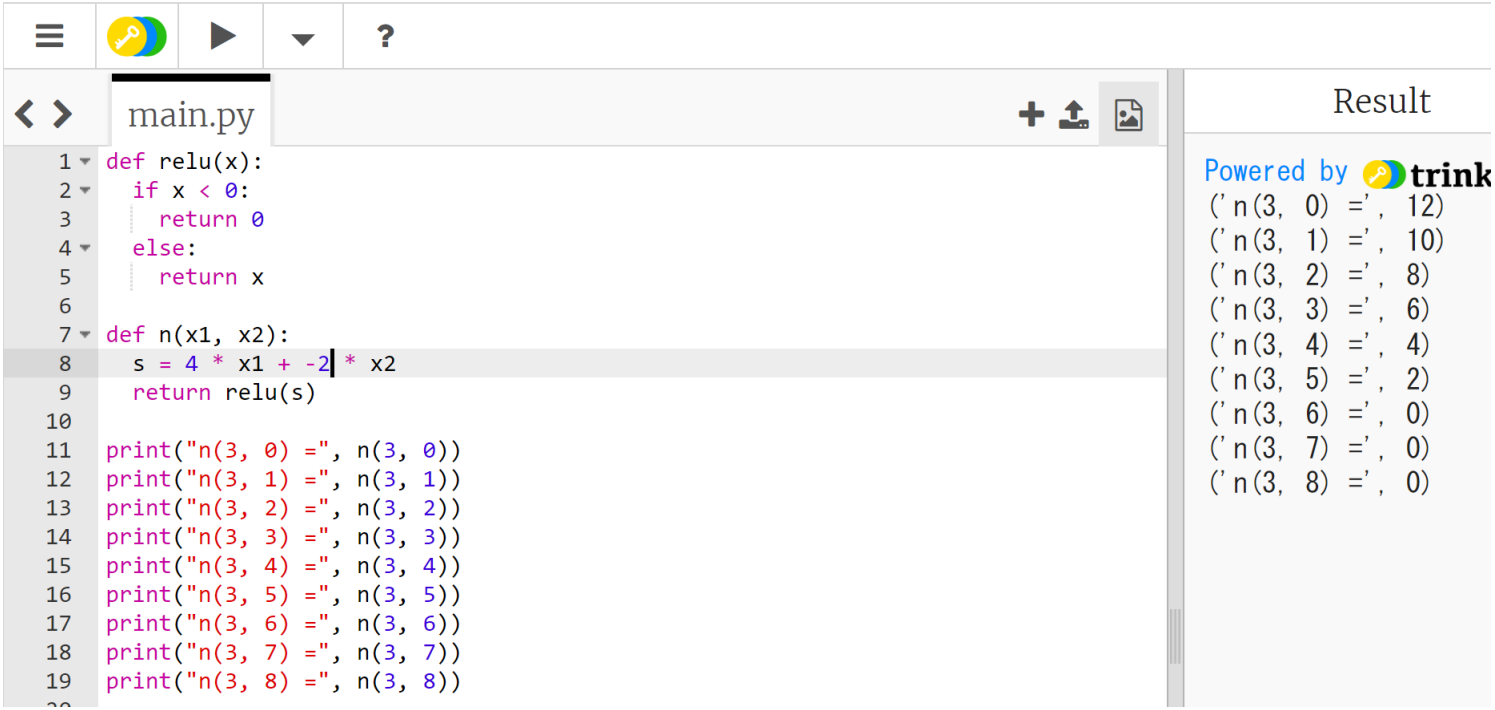


次のような**ニューロン**がある

- 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

ニューロンAの出力は3であるとする。
ニューロンBの出力が**2以下**のとき
　ニューロンCは**活性化**
ニューロンBの出力が少しでも**2を超えたとき**
　ニューロン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



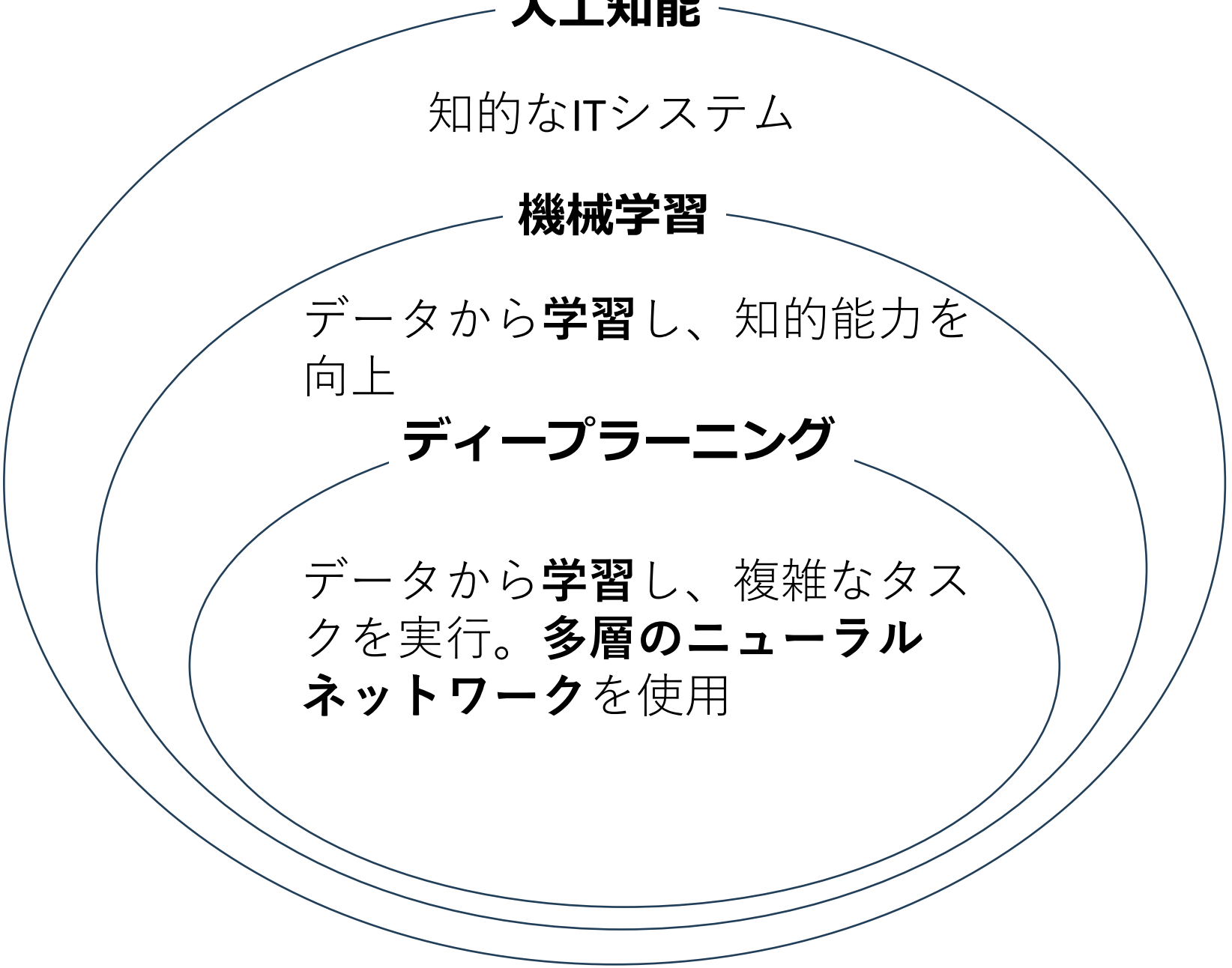
-2 を -6 に変えて、変化を見る

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
- **ニューロン**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つの層の間のニューロン**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．

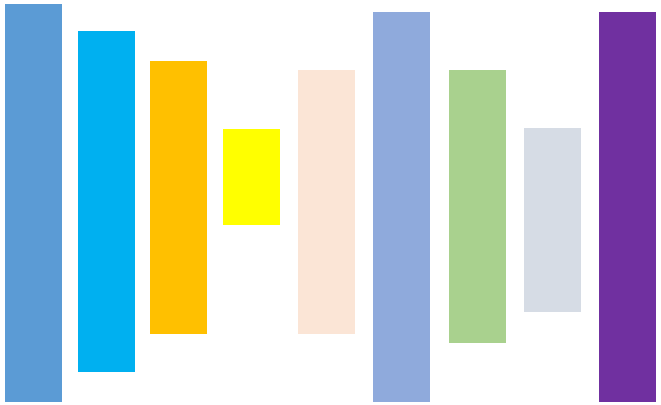
5-5. ディープラーニングの特徴と応用


人工知能
知的なITシステム
機械学習
データから学習し、知的能力を向上
ディープラーニング
データから学習し、複雑なタスクを実行。多層のニューラルネットワークを使用

ディープニューラルネットワーク

ディープラーニングは**多層のニューラルネットワーク**を用いた機械学習

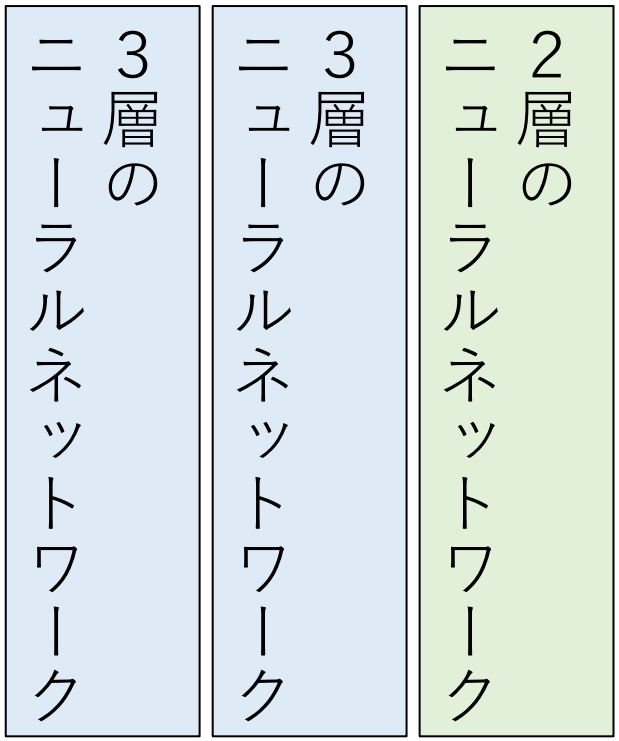
層の数が少ない（浅い）



層の数が多い（深い）

ディープニューラルネットワーク

- 層が浅い（**層の数が少ない**）**ニューラルネットワーク**を**組み合わせる**こともある



合計で８層

（さまざまな組み合わせがありえる）

ディープラーニングが広く利用されている理由

- **多様なデータへの対応**

例：画像、テキスト、音声、動画など

- **広範な応用分野**

例：画像認識、自然言語処理、音声認識など

- **優れたパターン抽出能力**

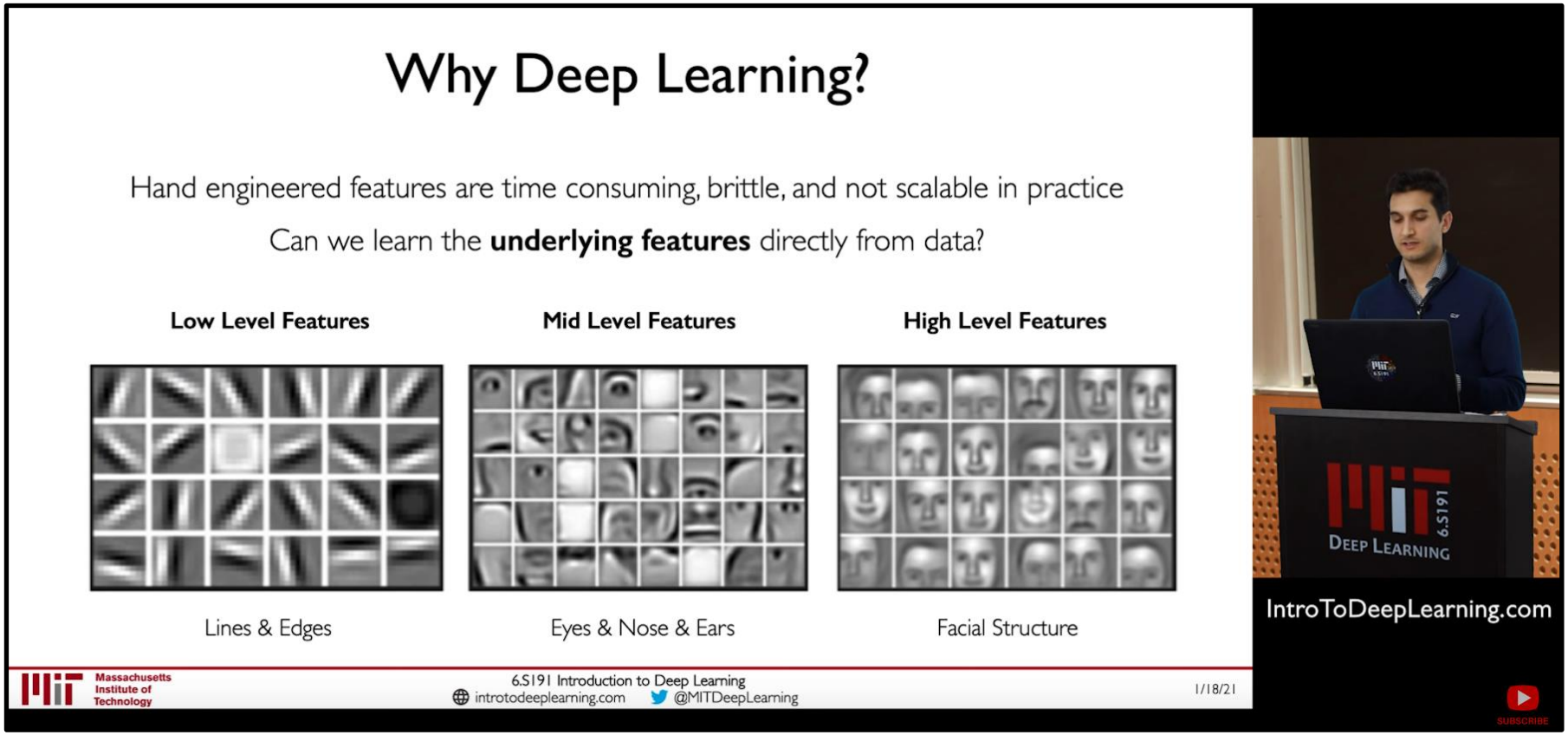
複雑なパターンを抽出する能力に優れる。

- **タスクの自動実行能力**

パターン抽出ならびにタスク実行は自動で行われる。

ディープラーニングへの期待

さまざまなレベルのパターン



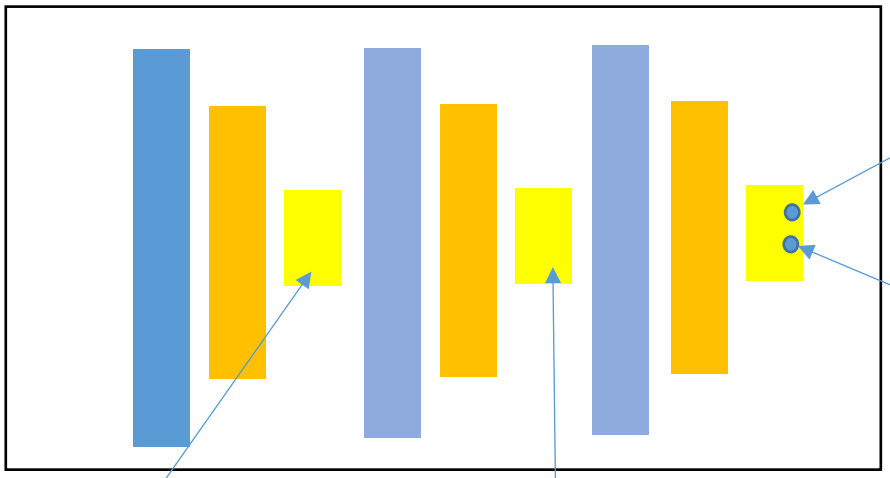
線や点のレベル　目，鼻，耳のレベル　顔の構造のレベル

MIT Introduction to Deep Learning | 6.S191,
https://www.youtube.com/watch?v=5tvmMX8r_OM
の「Why Deep Learning」のページ

教師なし学習のニュース (2011年)

- **教師なし学習**（この画像が「人の画像である」，「猫である」という**正解がない**）
- 訓練データ: YouTube から**ランダムに選ばれた画像 1000万枚**
- **1000台のマシンで，3日間の学習**
- **9層のニューラルネットワーク**を使用

高次のパターンを認識できる能力を獲得

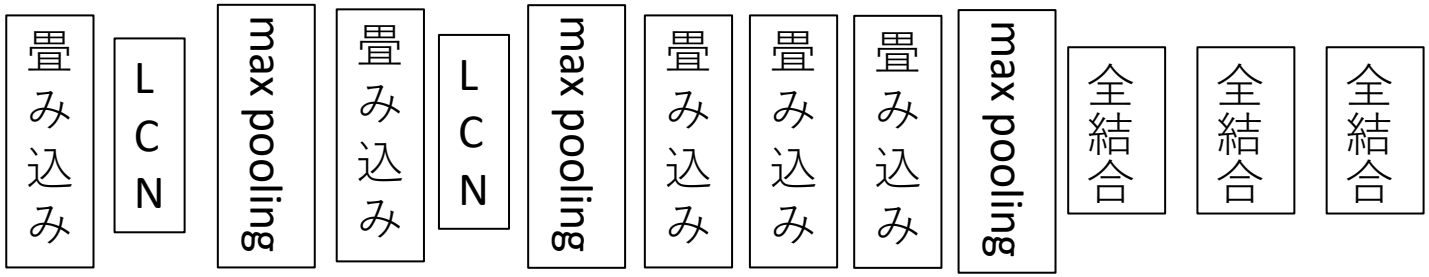


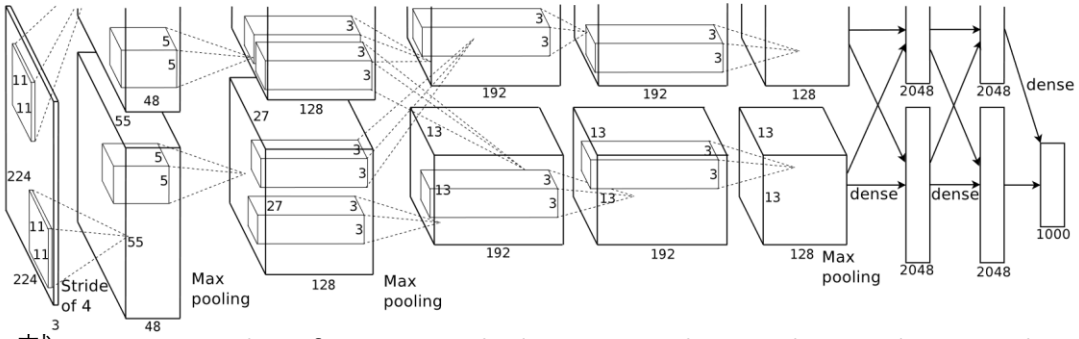
文献: Building high-level features using large scale unsupervised learning
Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, Andrew Y. Ng, arXiv 1112.6209, 2011, last revised 2012.

教師あり学習のニュース (2012年)

- **教師あり学習**の AlexNet で**画像分類**を行う
- **訓練データ: 画像約 100万枚以上**（ImageNet データセット, 22000種類に分類済み）
- ILSVRCコンペティション: 画像を **1000 種類**に**分類**
- **ディープニューラルネットワーク**を使用

畳み込み, max pooling, 正規化(LCN), softmax, ReLU, ドロップアウト

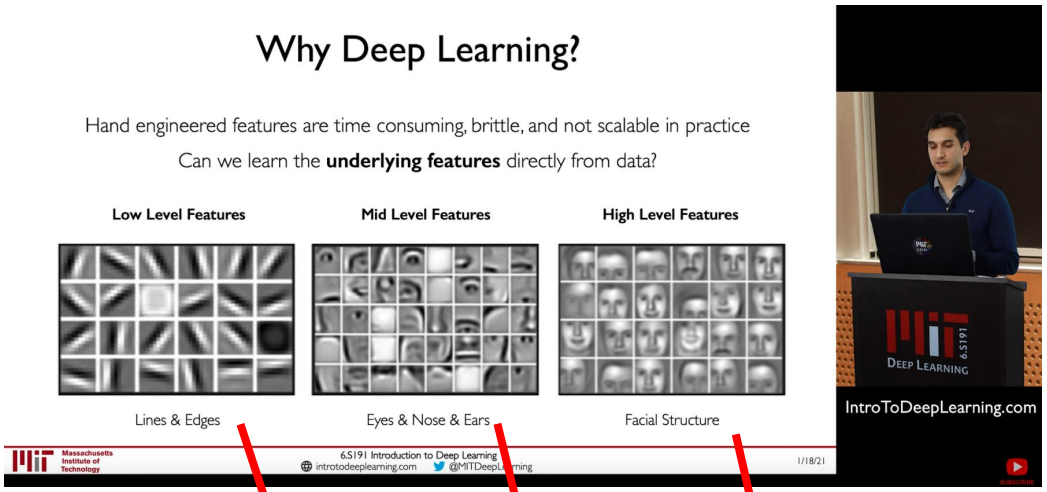


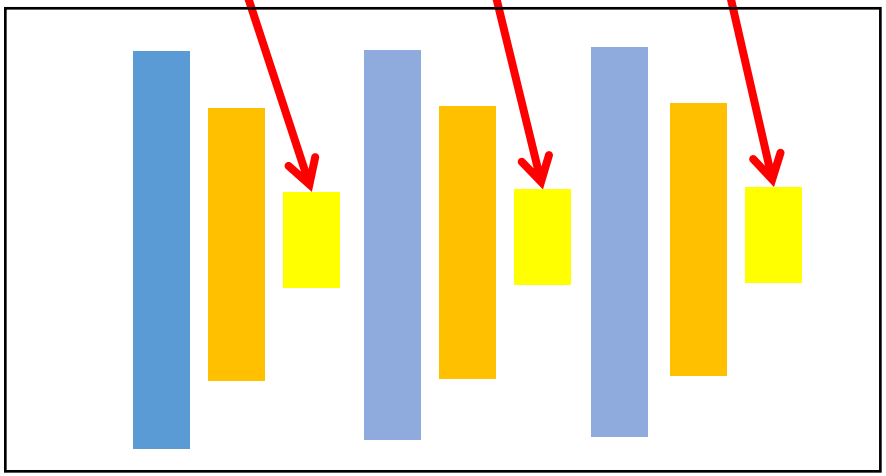


文献: ImageNet classification with deep convolutional neural networks, Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, NIPS'12, 2012.

ディープラーニングへの期待

さまざまなレベルのパターンを**抽出・認識**できるようになる」という考える場合も





「より出力に近い層では，**より高次のパターン**を認識したい」という考え方も

画像分類の精度の向上

ImageNet データセットの画像分類の結果

- **ディープラーニング**の進展
- **画像分類は，場合によっては，AIが人間と同等の精度**　とも考えられるように

画像分類の誤り率 (top 5 error)

人間: 5.1 %

PReLU による画像分類: 4.9 %

（2015年発表）

文献: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun,
Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification
arXiv:1502.01852, 2015.

ディープラーニングの応用と未来

ディープラーニングの研究は急速に進展

- 2011年：教師なし学習による顔認識の成功（YouTube画像1000万枚使用）
- 2012年：AlexNetによる画像分類の革新
- 2015年：人間の精度を上回る画像認識の達成（誤り率4.9%）

ディープラーニングを支える要素

- **活性化関数**（ReLUなど）
- **ドロップアウト**や**初期化手法**の開発
- **大規模データ**の活用
- **高性能コンピュータ**の発展
- **データ拡張技術**の進歩

ディープラーニングの応用と特徴

- **ディープラーニング**は**機械学習**の一種であり、人工**ニューラルネットワーク**を使用して**データから学習**し、複雑な**タスクを実行**する技術

- 「ディープ」の名前は、**多層のニューラルネットワーク**を使用することに由来

- ディープラーニングが広く利用される理由は、**多様なデータに適用**でき、**さまざまなタスク**で高性能を発揮するため。例えば、**画像認識**、**自然言語処理**、**音声認識**など。

自習１．「**１．活性化関数 ReLU**」について

目的: この自習問題の目的は、**活性化関数 ReLU**がどのように動作するかを理解し、プログラム内で実際に実行してみることです。

- プログラムを読んで、それが何をしているのか理解してください。
- **relu 関数がどのように動作するか**を特に注意して理解しましょう。
- **プログラムを書き換えて、-3 や 3 のときの ReLU の値を計算**させてみましょう。

ヒント:

- relu 関数は、与えられた x が負の場合は 0 を返し、それ以外の場合は x 自体を返します。
- x の値を変更して、異なる x の値に対する ReLU の値を計算させてみてください。-3 や 3 などを試してみましょう。

　ReLU 関数は、ニューラルネットワークで広く使用されます。そのため重要です。

自習は提出する必要はありません。

自習２．「**２．ニューロン、重み、バイアス、活性化関数、ニューロンの活性度**」について

目的:ニューラルネットワークのニューロンがどのように入力に対して活性度を計算し、ReLU 活性化関数を使用して活性度を処理するかを理解することです。

- プログラムのコードを読んで、それが何をしているのか理解してください。n 関数は、入力、重み、バイアス、活性化関数を用いて、活性度を計算するためのものです。
- **プログラムを書き換えて、異なる入力値 (0.1, 0.8, -0.5)、(5.1, 0.8, -0.5)、(10.1, 0.8, -0.5)、(15.1, 0.8, -0.5) を試してみましょう。**

ヒント:

- n 関数内で、s は重み付きの入力とバイアスを用いて計算された値です。この値を relu 関数に渡して、活性度を求めています。

ニューロンの活性度の計算は、ニューラルネットワークの重要な概念です

自習は提出する必要はありません。