

aa-5. 深層学習 (ディープラーニング)

(人工知能)

URL: <https://www.kkaneko.jp/ai/mi/index.html>

金子邦彦





① ニューラルネットワークの基本的仕組み、から深層学習（ディープラーニング）の概要、分類への応用、学習の原理まで、必要な事項を体系的に説明。

② ニューロンの働き、層構造、結合の重みの役割を図を交えてビジュアルに分かりやすく説明。イメージしやすい。

③ 原理の説明と、例での説明の両方により、ニューラルネットワークの実際の応用と本質を理解。

まとめや確認問題で理解を確実に。易しい内容から難しい内容まで幅広く説明。

アウトライン

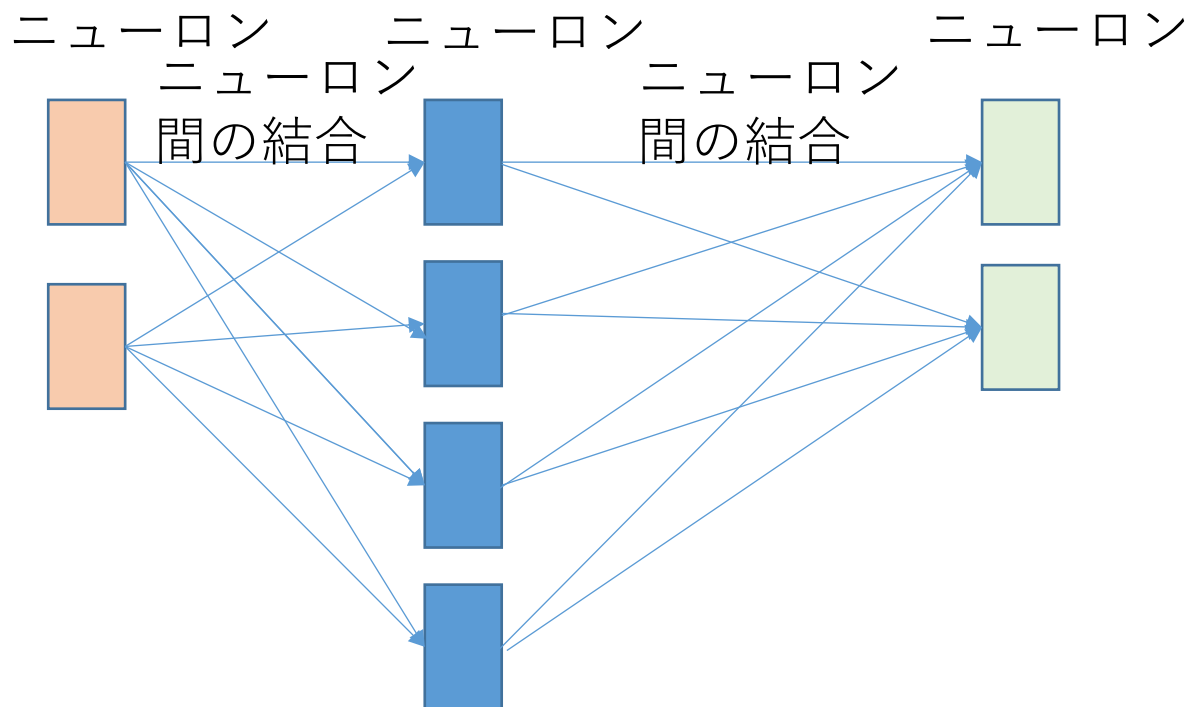
1. ニューラルネットワークの基本構造
2. 活性化関数とその役割
3. フォワードプロパゲーション
4. ディープラーニングの特徴と応用
5. ニューラルネットワークを用いた分類

5-1. 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



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



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: 各入力値に対して，それぞれの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: 合計を計算し，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．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）

バイアス 0.2

$$0.3 \times 0.1 \Rightarrow 0.03$$

$$-0.5 \times 0.8 \Rightarrow -0.4$$

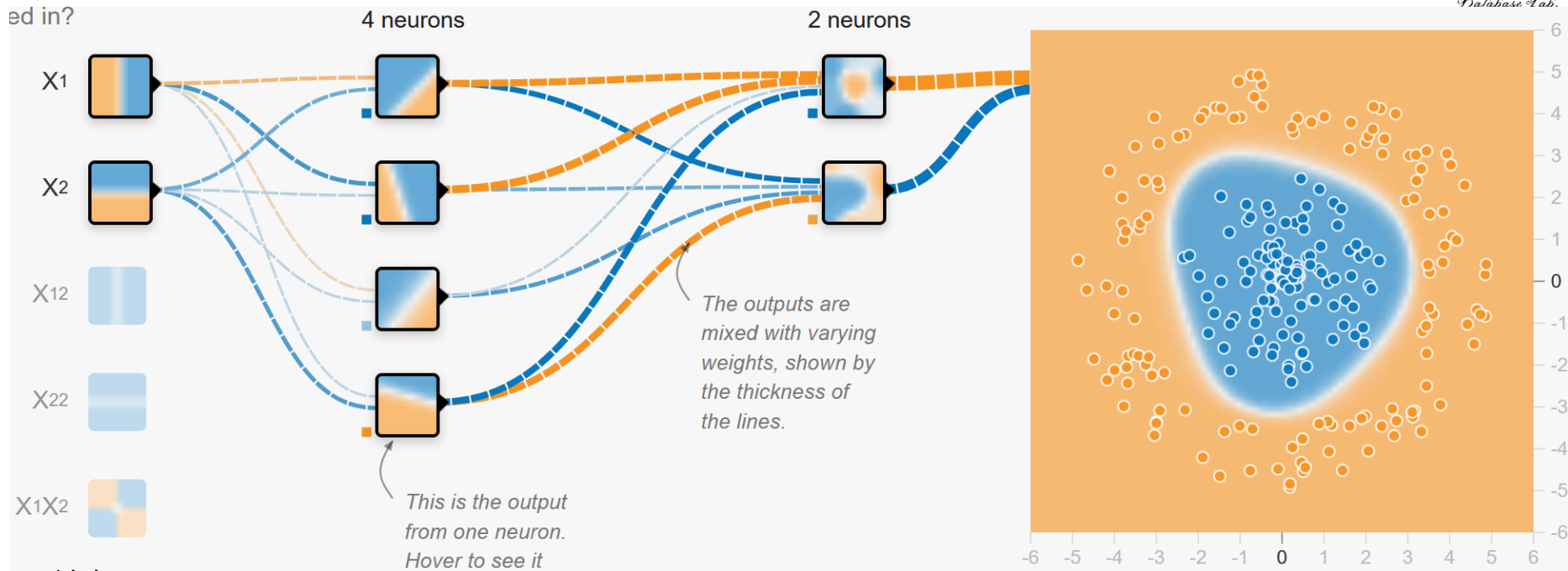
$$0.2 \times -0.5 \Rightarrow -0.1$$

入力 重み

合計
 -0.47



-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

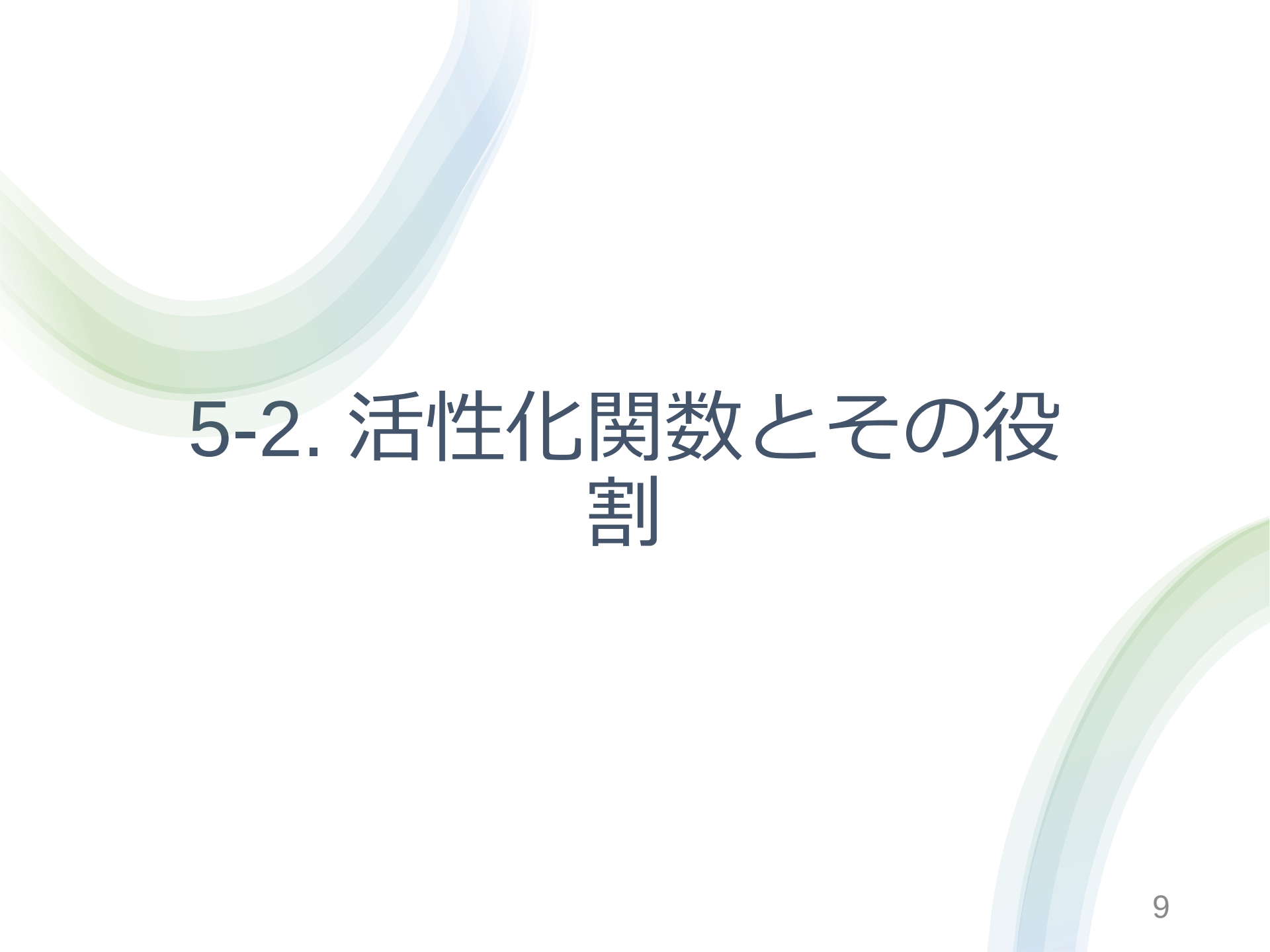
→結合→ 1 層目 →結合→ 2 層目

ニューラルネットワーク

ニューロンの活性化と非活性化

ニューロンの出力が、プラスの大きな値
→ **活性化**している

ニューロンの出力が、0に近い値
→ **活性化**していない

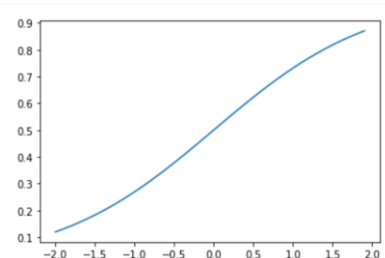


5-2. 活性化関数とその役割

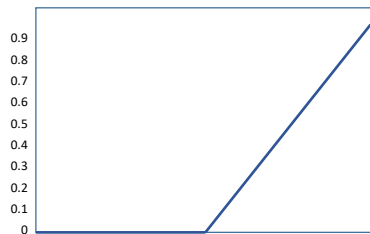
活性化関数



- 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

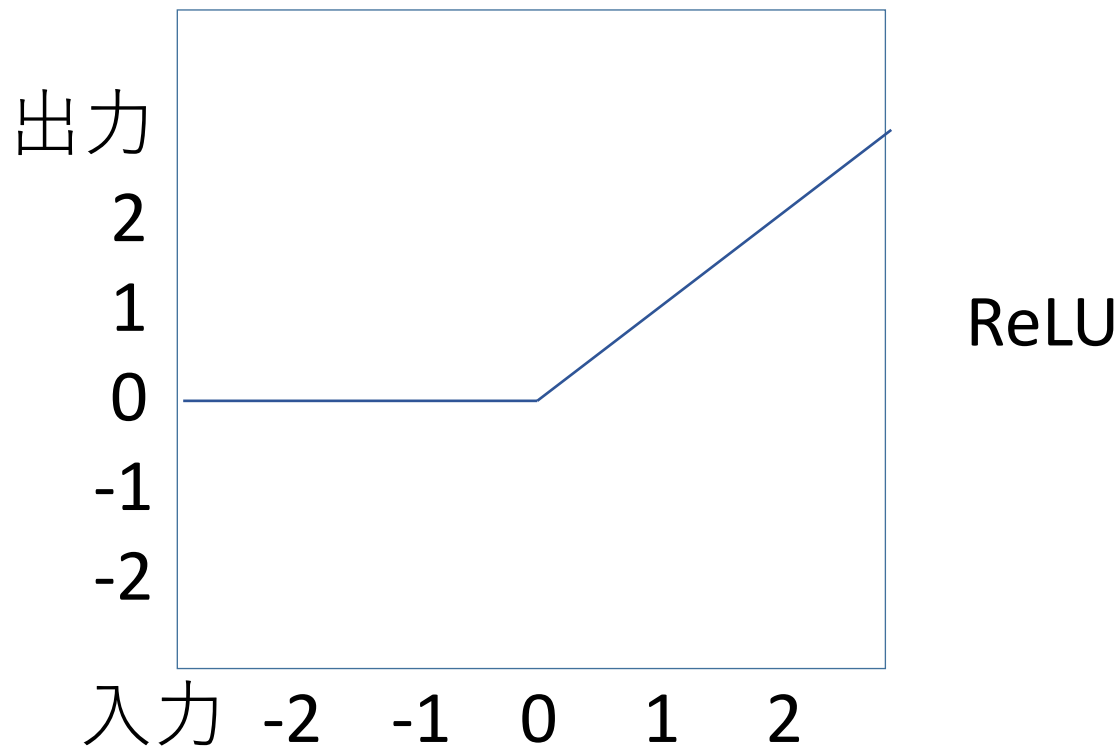


シグモイド



ReLU

- 2011年に発表された活性化関数
- 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
- シンプルで，性能が良く，実績が豊富で，扱いやすい



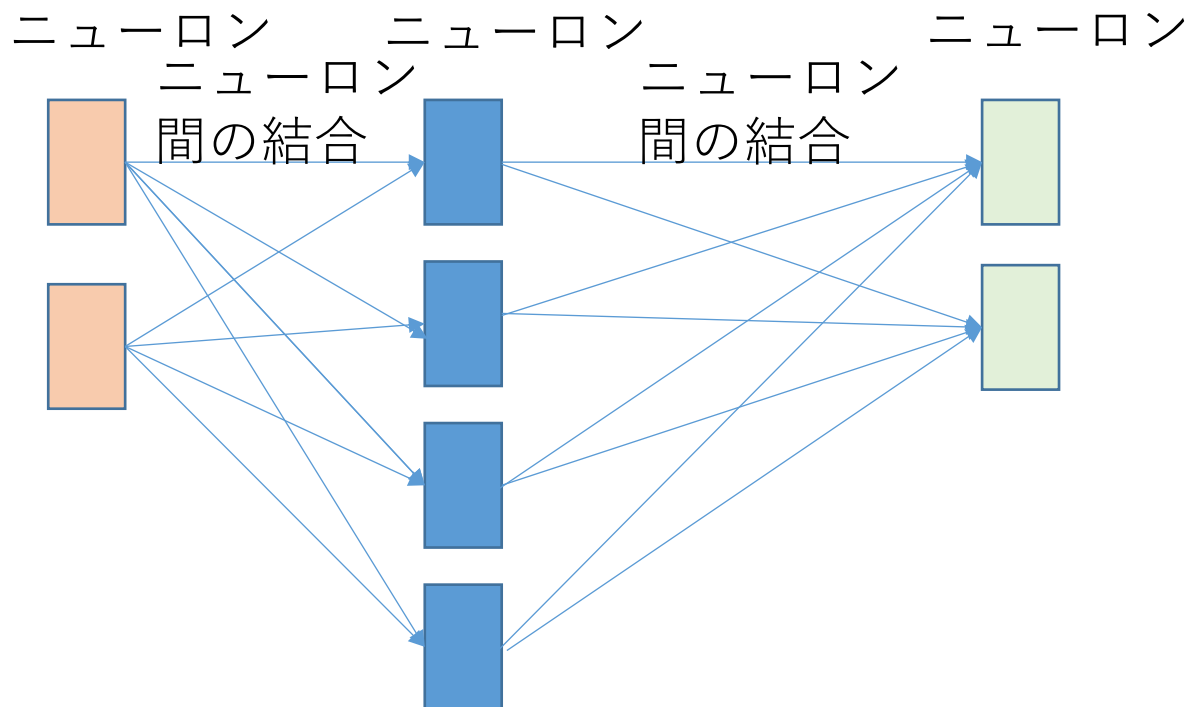


5-3. フォワードプロパ ゲーション

ニューラルネットワークの仕組み

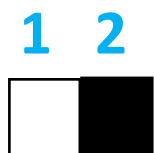


入力の重みづけ, 合計とバイアス, 活性化関数の適用を行う
ニューロンがネットワークを形成



①入力

数値，複数可 例： 1 0

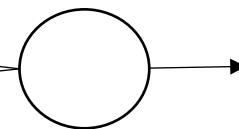


入力

入力

1 1

2 0



ニューロン

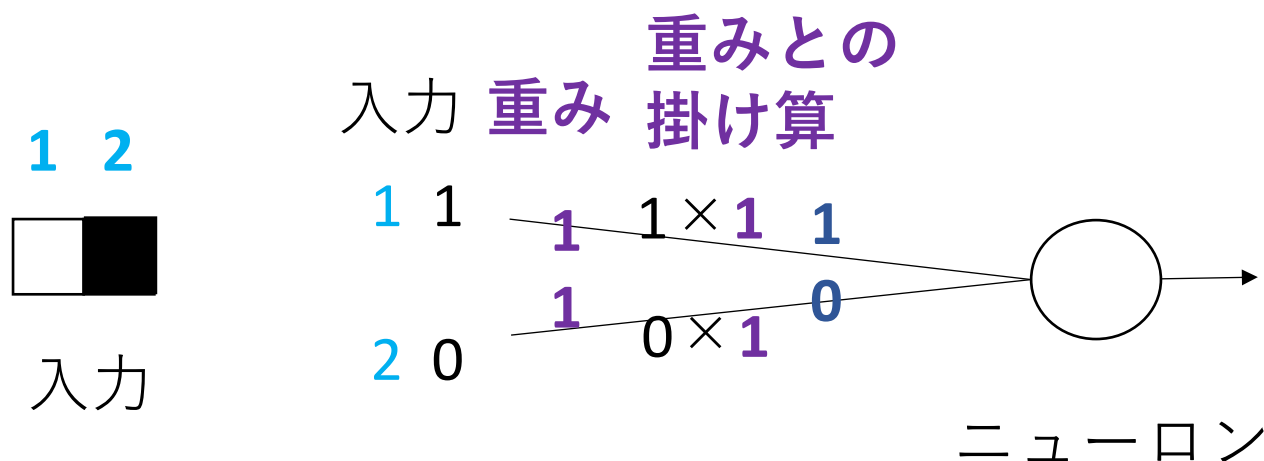
白を 1
黒を 0 とする

②入力の重みづけ

- 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

- 重み**は、**入力データの重要度**を決定（**学習過程で自動的に調整**され、望ましい出力が得られるように最適化される）



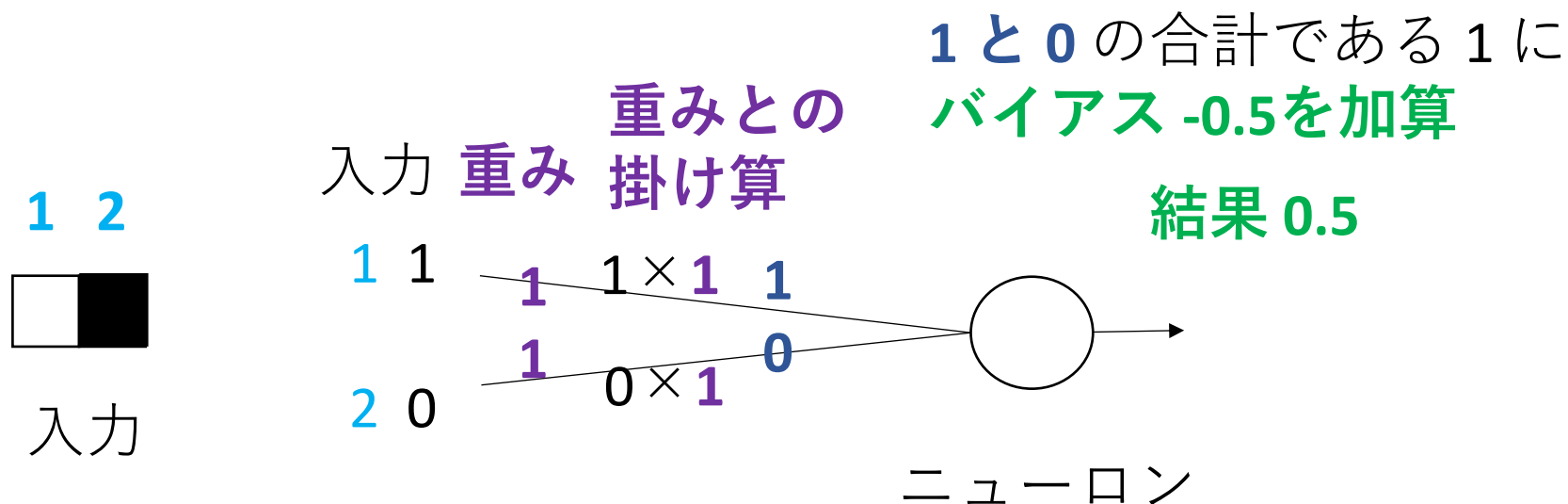
白を 1
黒を 0 とする

③ バイアス

- 合計にバイアス（数値）を加える.

例 10 の合計は 1. これに -0.5 を加える

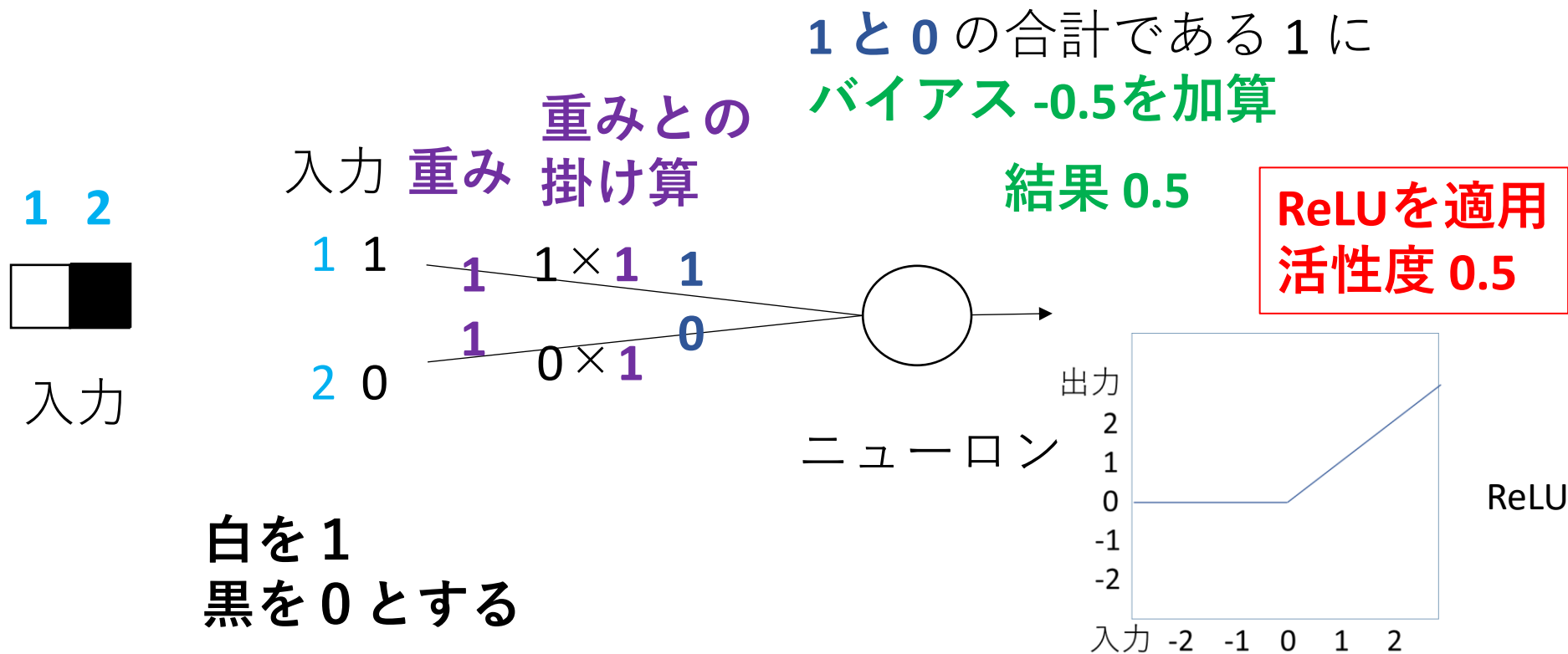
- バイアスは、ニューロンの活性化しやすさを決定（学習過程で自動的に調整され、望ましい出力が得られるように最適化される）



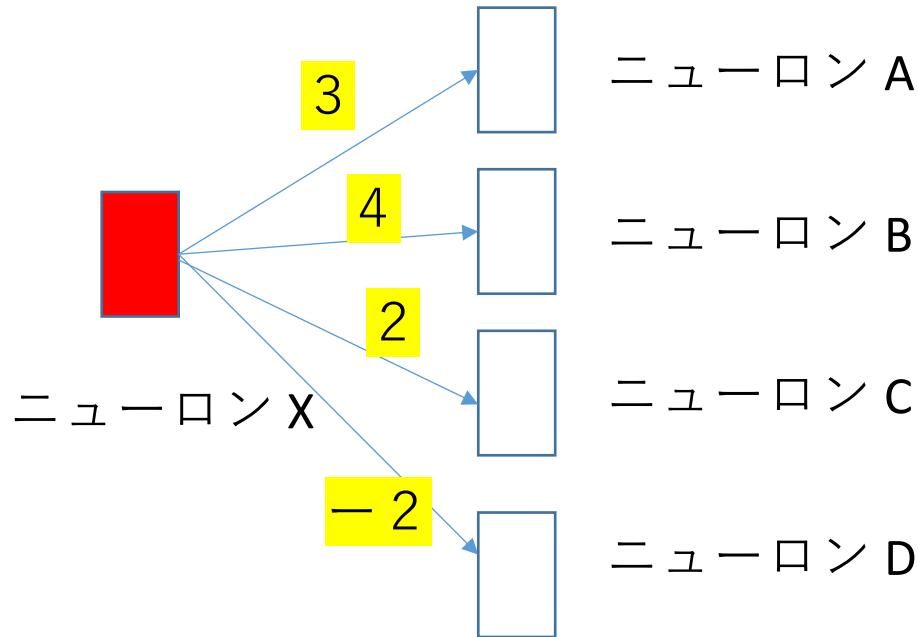
白を 1
黒を 0 とする

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



次のニューロンの入力
= **前のニューロンの出力**
× **結合の重み**

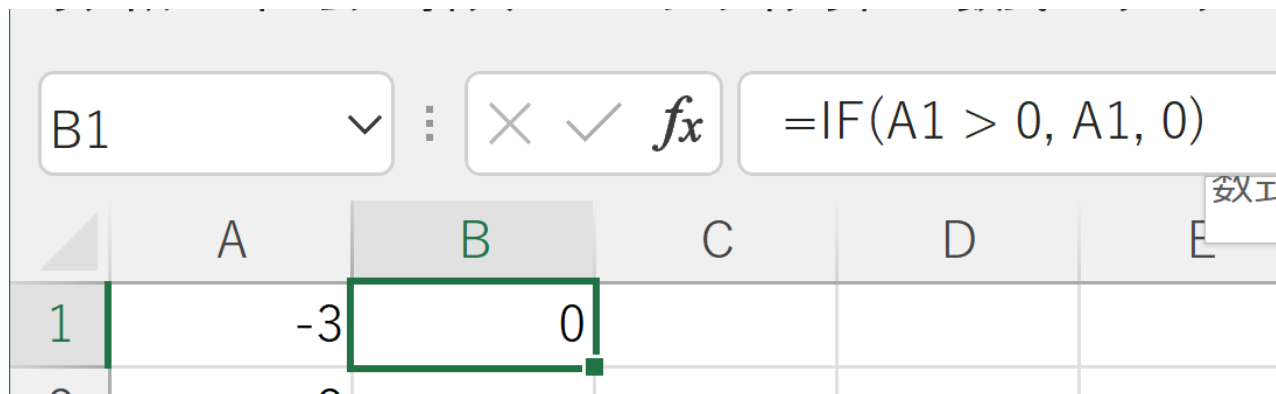
結合の重みは、
結合ごとに違う値

活性化関数 ReLU

0 以下の場合 → 結果は 0

0 以上の場合 → 結果は, そのままの値

Excel では, 次のような式になる
式「**=IF(A1 > 0, A1, 0)**」



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E
1	-3	0			

The formula bar at the top shows the formula `=IF(A1 > 0, A1, 0)` entered into cell B1. The cell B1 is highlighted with a green border, and the value 0 is displayed inside it. The formula bar also shows the function icon f_x and the cell reference B1.

Excel を起動. 起動したら「空白のブック」を選ぶ



Online template search (オンライン テンプレートの検索)

Search results: Business (ビジネス), Budget (予算), Calendar (カレンダー), Overview (一覧), Personal (個人用), Small Business (小規模ビジネス), Tablet (電卓)

Blank Workbook (空白のブック)

Excel へようこそ

毎日の作業スケジュール

従業員の出勤簿

従業員のシフトのスケジュール

① セル A1 から A7 に, 値 -3, -2, -1, 0, 1, 2, 3
(半角で)

A7			
✕ ✓ f_x 3			
	A	B	C
1	-3		
2	-2		
3	-1		
4	0		
5	1		
6	2		
7	3		

② ReLU の式

セル **B1** に式「**=IF(A1 > 0, A1, 0)**」

CORREL ▾ : ✕ ✓ f_x =IF(A1 > 0, A1, 0)					
	A	B	C	D	E
1	-3	=IF(A1 > 0, A1, 0)			
2	-2				
3	-1				
4	0				
5	1				
6	2				
7	3				

③ ReLU の式
セル **B1** の式を,
B2 から B7 に「コピー&貼り付け」
右クリックメニューが便利

	A	B	C	D	E
1	-3	0			
2	-2	0			
3	-1	0			
4	0	0			
5	1	1			
6	2	2			
7	3	3			

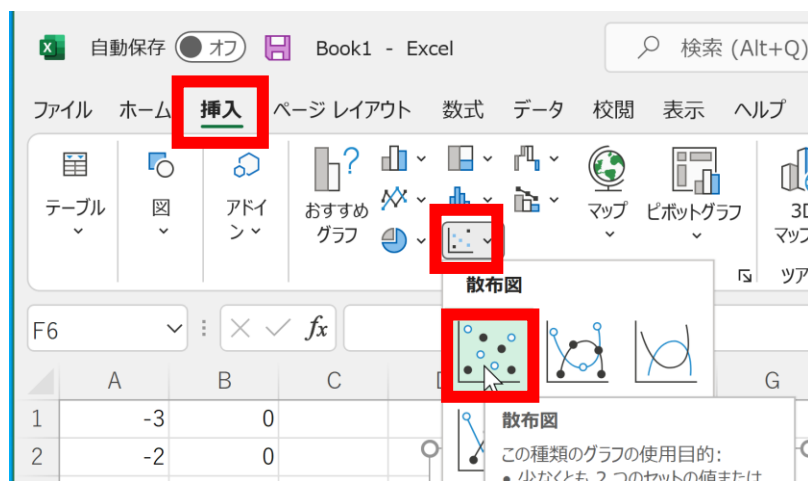
④ 散布図の作成

	A	B
1	-3	0
2	-2	0
3	-1	0
4	0	0
5	1	1
6	2	2
7	3	3

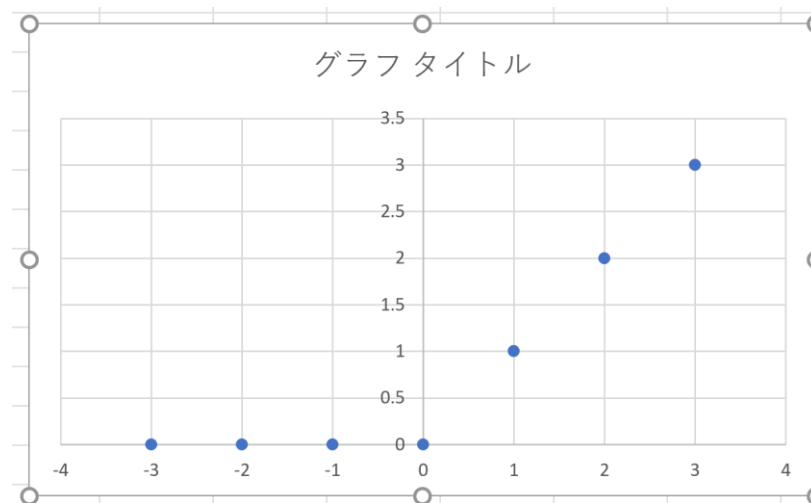
元データ

	A	B
1	-3	0
2	-2	0
3	-1	0
4	0	0
5	1	1
6	2	2
7	3	3

① グラフ化したい部分を範囲選択

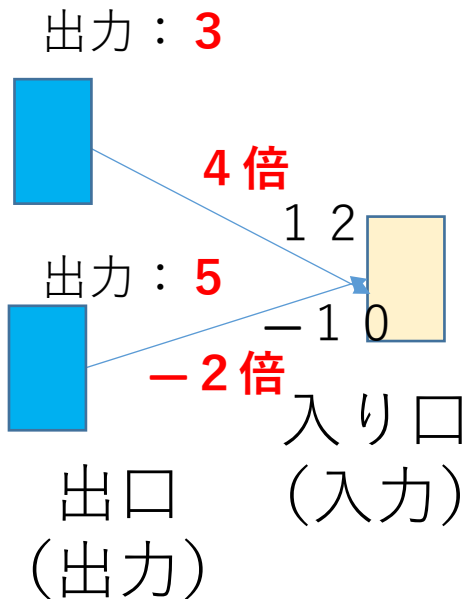


② リボンで「挿入」→散布図の選択



散布図が得られる

⑤ 前のニューロンの出力と，結合の重み
セル A8, A9, B8, B9 に次の値を入れる



8	3	4
9	5	-2

⑥ (入力) × (結合の重み) の合計

- ・セル C8 に, 式「=A8*B8」
- ・セル C9 に, 式「=A9*B9」
- ・セル C10 に, 式「=SUM(C8:C9)」

8	3	4	12
9	5	-2	-10
10			2

ここではバイアスは0と考える

⑦求まった合計に**活性化関数 ReLU** を適用
→ ニューロンの出力になる

・セル C11 に, 式「**=IF(C10 > 0, C10, 0)**」

CORREL ✕ ✓ fx =IF(C10 > 0, C10, 0)					
	A	B	C	D	E
1	-3	0			
2	-2	0			
3	-1	0			
4	0	0			
5	1	1			
6	2	2			
7	3	3			
8	3	4	12		
9	5	-2	-10		
10			2		
11			=IF(C10 > 0, C10, 0)		
12					

演習 ReLU (Pythonで実現)

- **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



The screenshot shows the Trinket.io Python editor interface. On the left, a code editor displays a Python script for a ReLU function. The code defines a `relu(x)` function that returns 0 for negative values and `x` for non-negative values. It then tests the function with values -2, -1, 0, 1, and 2, printing the results. On the right, the 'Result' panel shows the output of the script, which is a list of strings showing the input `x` and the corresponding `relu(x)` result for each test case.

```
1 # ReLU (Rectified Linear Unit) 関数の実装と動作確認のプログラムです。
2 # 入力値xが0未満の場合は0を返し、0以上の場合はxをそのまま返します。
3 # -2から2までの異なる入力値でReLU関数を呼び出し、その変換結果を表示
4 # することで、活性化関数としてのReLUの基本的な挙動を示しています。
5
6 def relu(x):
7     if x < 0:
8         return 0
9     else:
10        return x
11
12 x = -2
13 print("x = ", x, "relu(x) =", relu(x))
14 x = -1
15 print("x = ", x, "relu(x) =", relu(x))
16 x = 0
17 print("x = ", x, "relu(x) =", relu(x))
18 x = 1
19 print("x = ", x, "relu(x) =", relu(x))
20 x = 2
21 print("x = ", x, "relu(x) =", relu(x))
```

Result

Powered by  trinket

```
('x = ', -2, 'relu(x) =', 0)
('x = ', -1, 'relu(x) =', 0)
('x = ', 0, 'relu(x) =', 0)
('x = ', 1, 'relu(x) =', 1)
('x = ', 2, 'relu(x) =', 2)
```

- 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 入力の重みづけ, 合計 とバイアス, 活性化関数の 適用

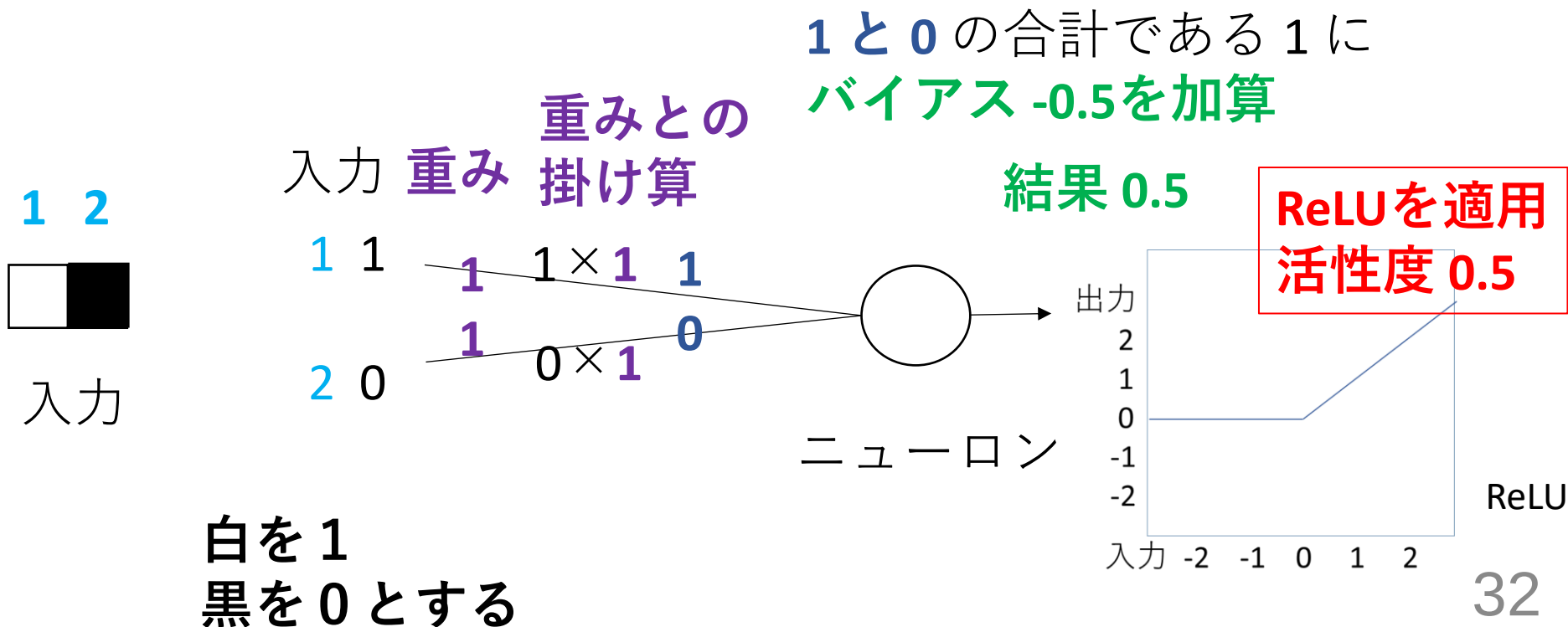


演習の狙い



- **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ます。
- 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ります。

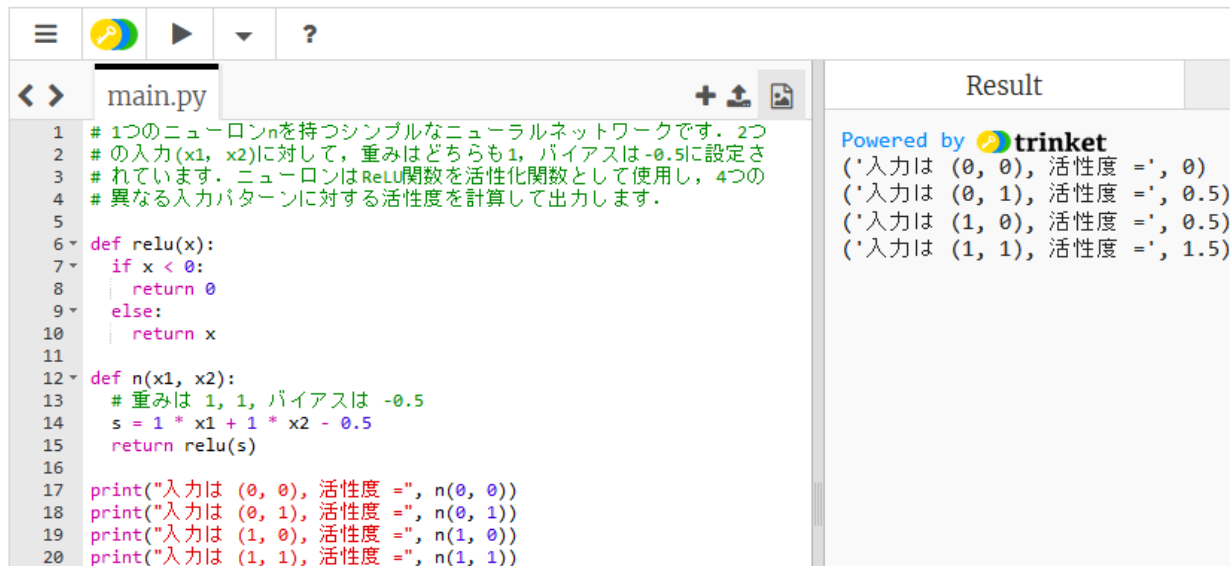
【入力が 1 0 のとき】



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



```
1 # 1つのニューロンnを持つシンプルなニューラルネットワークです。2つ
2 # の入力(x1, x2)に対して、重みはどちらも1, バイアスは-0.5に設定さ
3 # れています。ニューロンはReLU関数を活性化関数として使用し、4つの
4 # 異なる入力パターンに対する活性度を計算して出力します。
5
6 def relu(x):
7     if x < 0:
8         return 0
9     else:
10        return x
11
12 def n(x1, x2):
13     # 重みは 1, 1, バイアスは -0.5
14     s = 1 * x1 + 1 * x2 - 0.5
15     return relu(s)
16
17 print("入力は (0, 0), 活性度 =", n(0, 0))
18 print("入力は (0, 1), 活性度 =", n(0, 1))
19 print("入力は (1, 0), 活性度 =", n(1, 0))
20 print("入力は (1, 1), 活性度 =", n(1, 1))
```

Result

Powered by  trinket

('入力は (0, 0), 活性度 =' , 0)
('入力は (0, 1), 活性度 =' , 0.5)
('入力は (1, 0), 活性度 =' , 0.5)
('入力は (1, 1), 活性度 =' , 1.5)

- 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:

$1 \times 1, 1 \times 1, 1 \times 1, 0 \times 0, 1 \times 1, 1 \times 1, 0 \times 0, 0 \times 0, 1 \times 1$

2. 合計とバイアス:

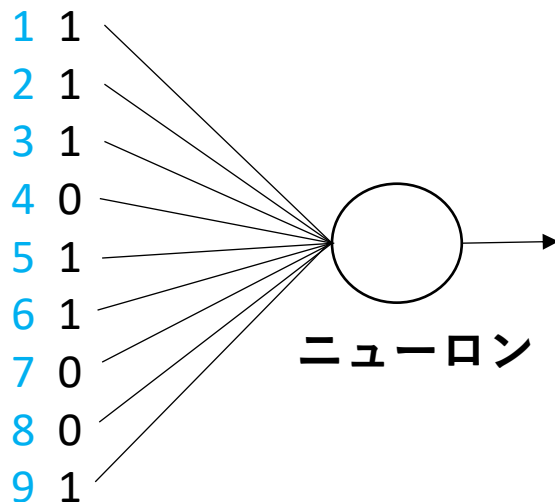
合計 6 にバイアス -5 を足す. 結果 1

3. 活性化関数の適用

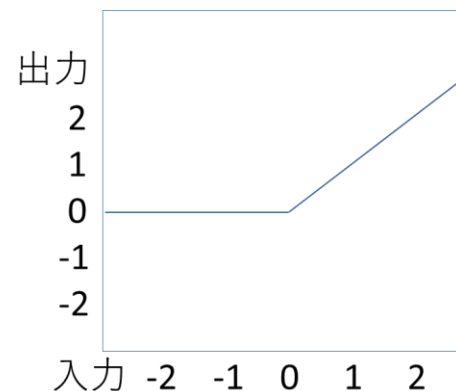
結果 1 に ReLU を適用した場合は, 活性度 1

1	2	3
4	5	6
7	8	9

入力



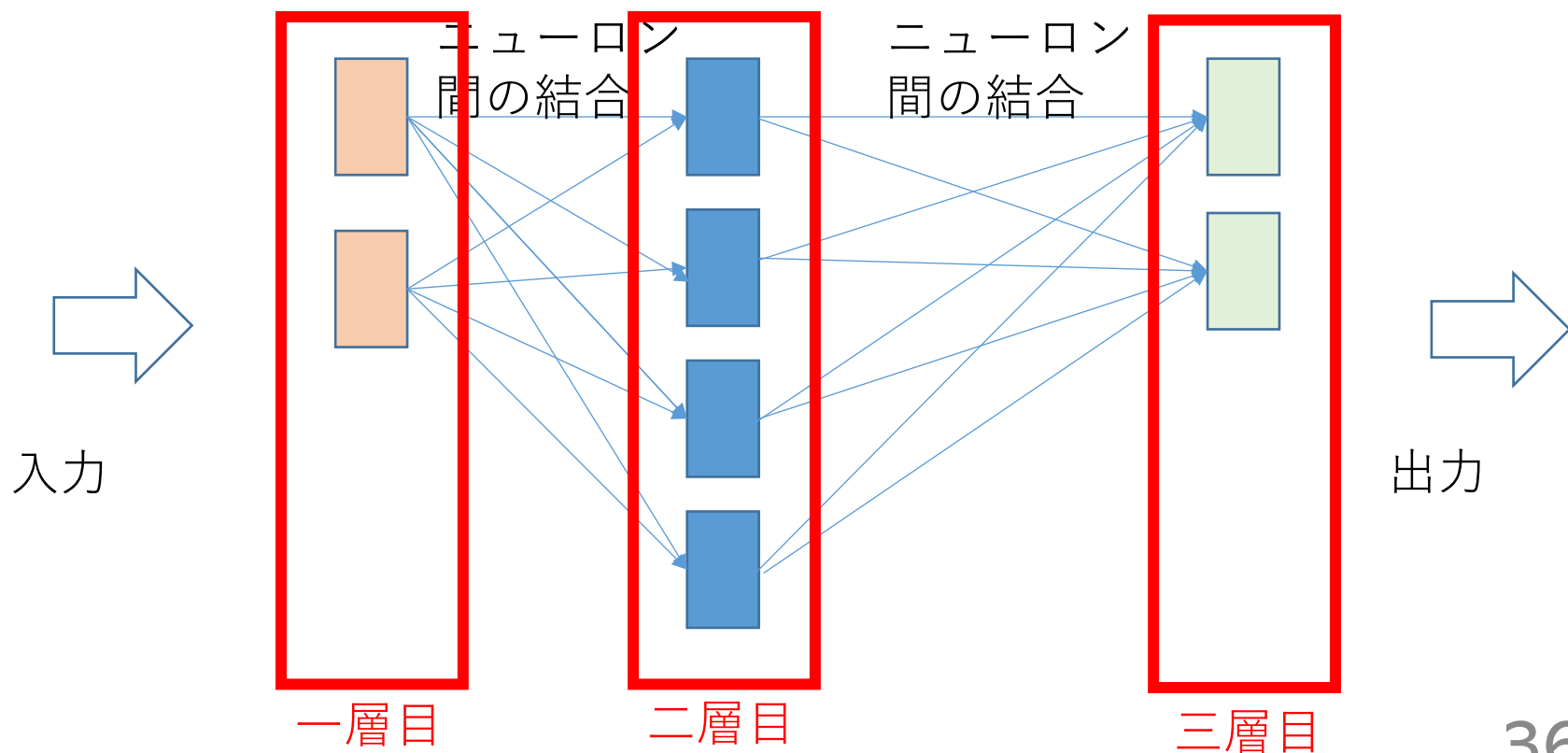
ReLU



白黒の画像
(画素は 0 または 1)

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．

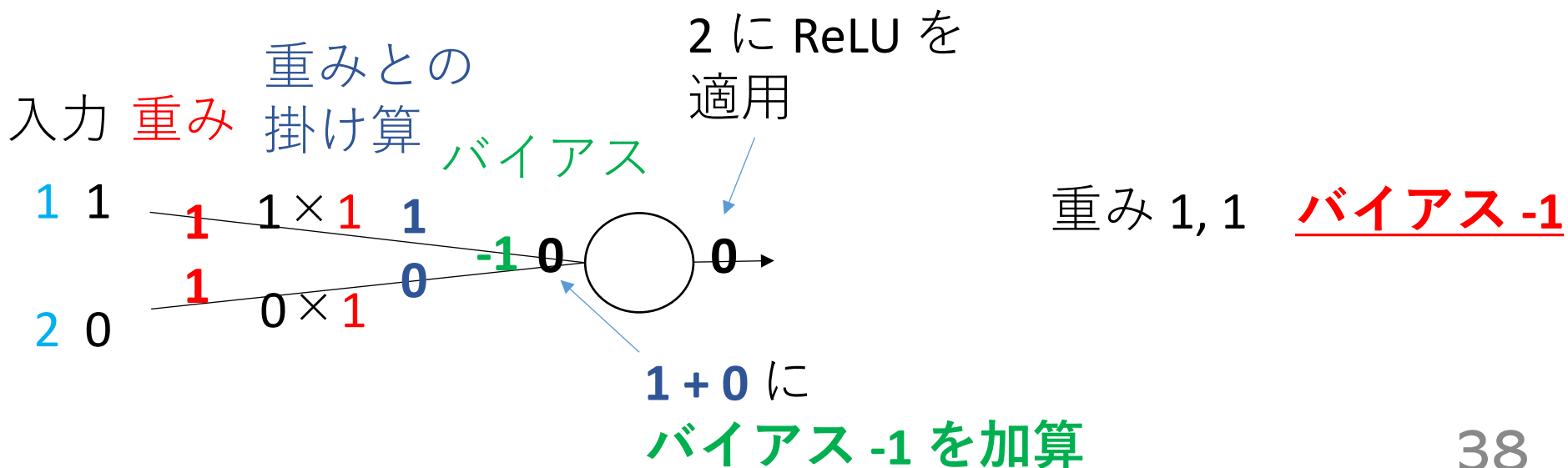
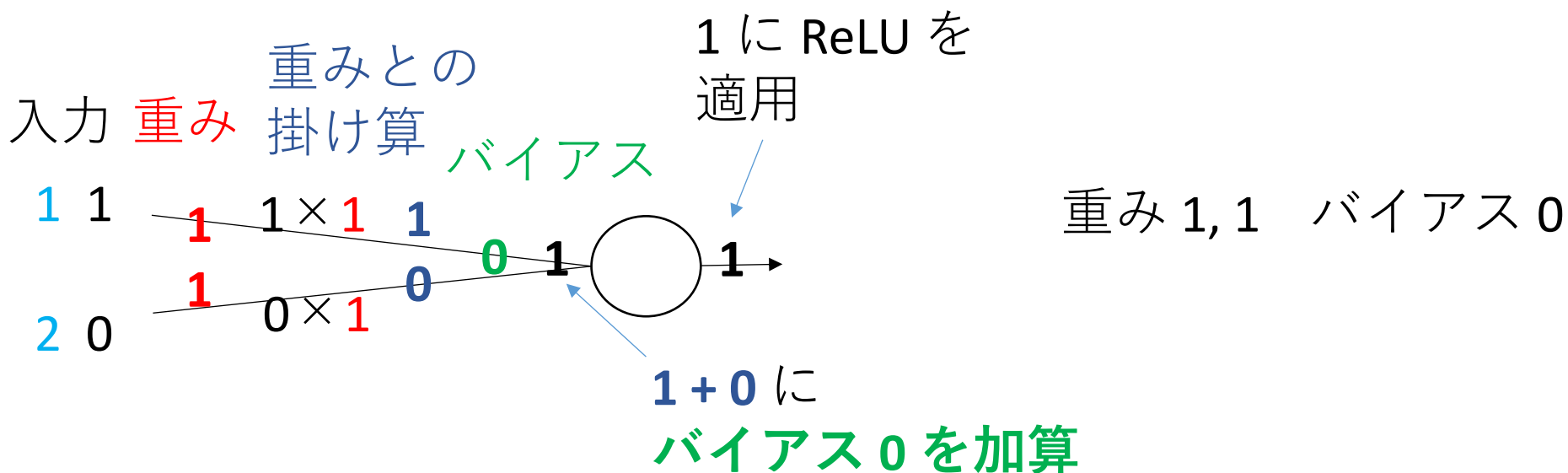


フォワードプロパゲーション

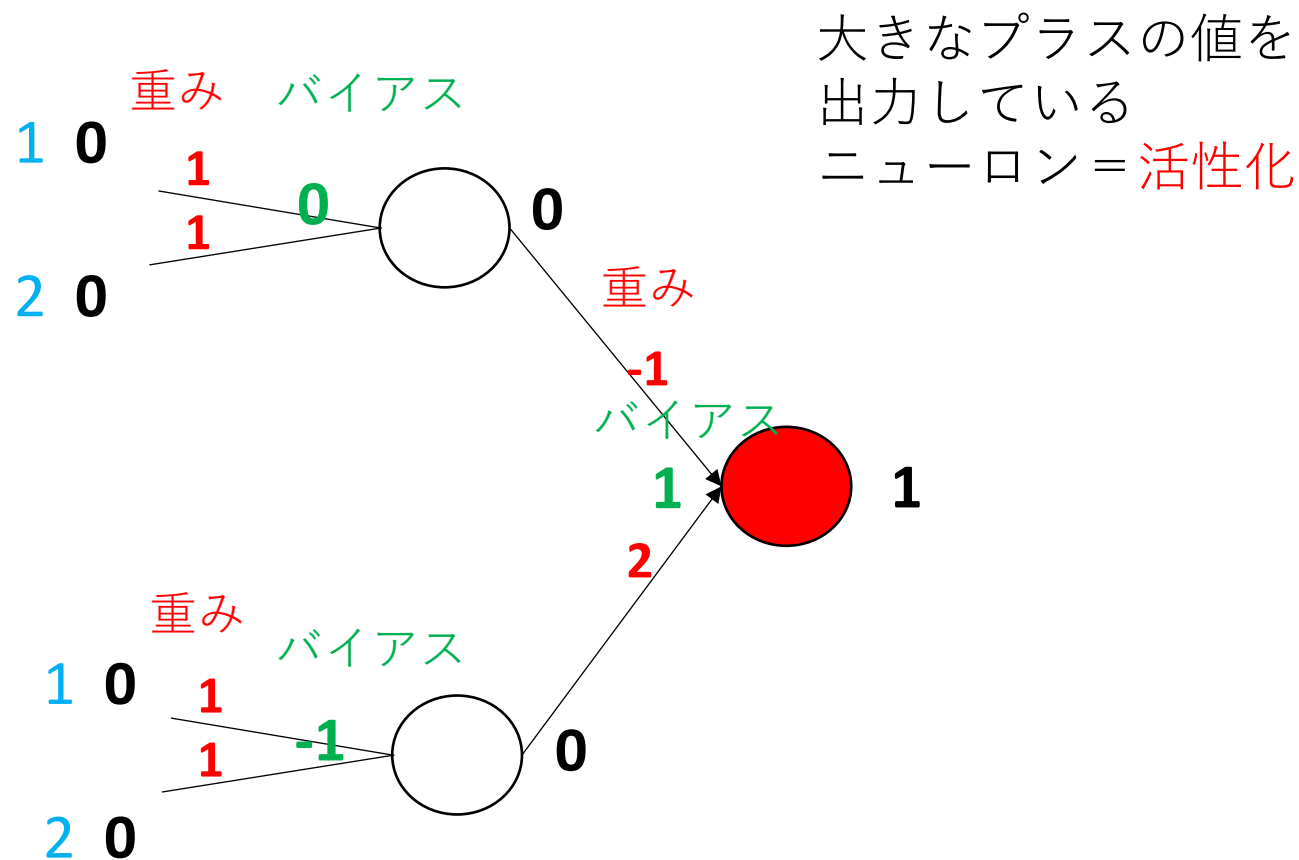


- **フォワードプロパゲーション**は、**ニューラルネットワーク**において、**データが入力層から出力層へ順方向に伝播**する仕組み。
- **各ニューロン**は**前の層から受け取ったデータ**を処理し、その結果を**次の層に伝達**する。

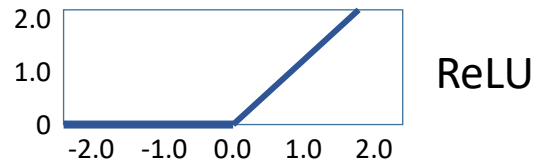
パラメータ（バイアス）の変化による出力の変化

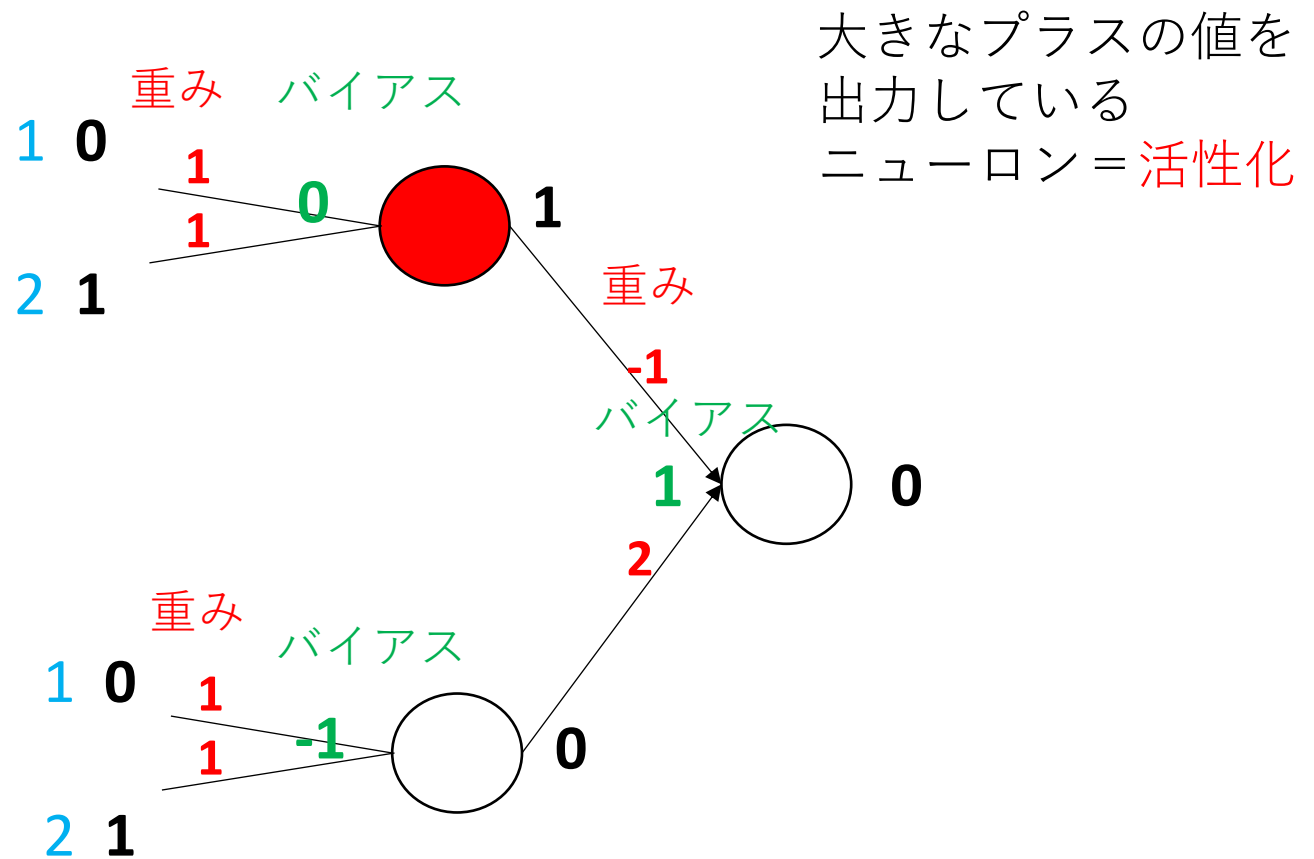
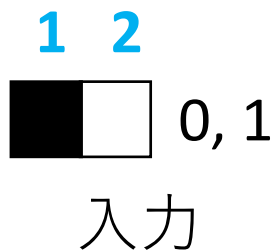


1 2
0, 0
入力

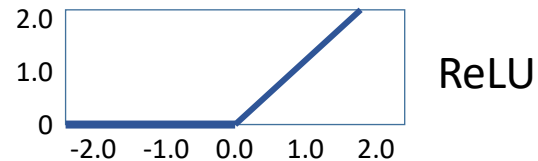


3つのニューロンの活性化関数はすべて ReLU

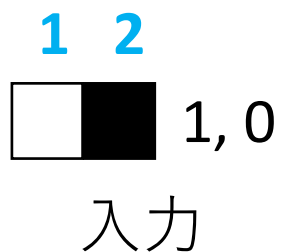
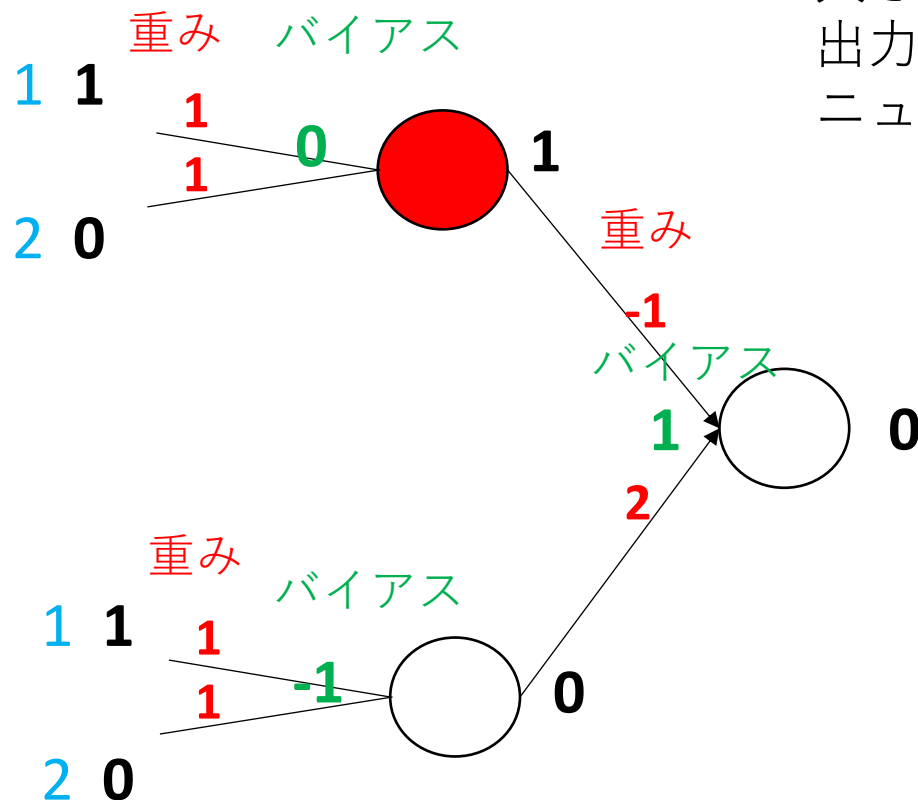




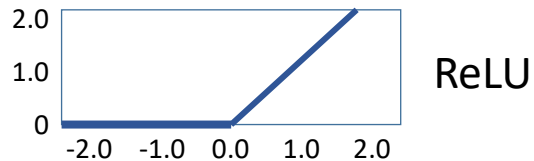
3つのニューロンの活性化関数はすべて ReLU



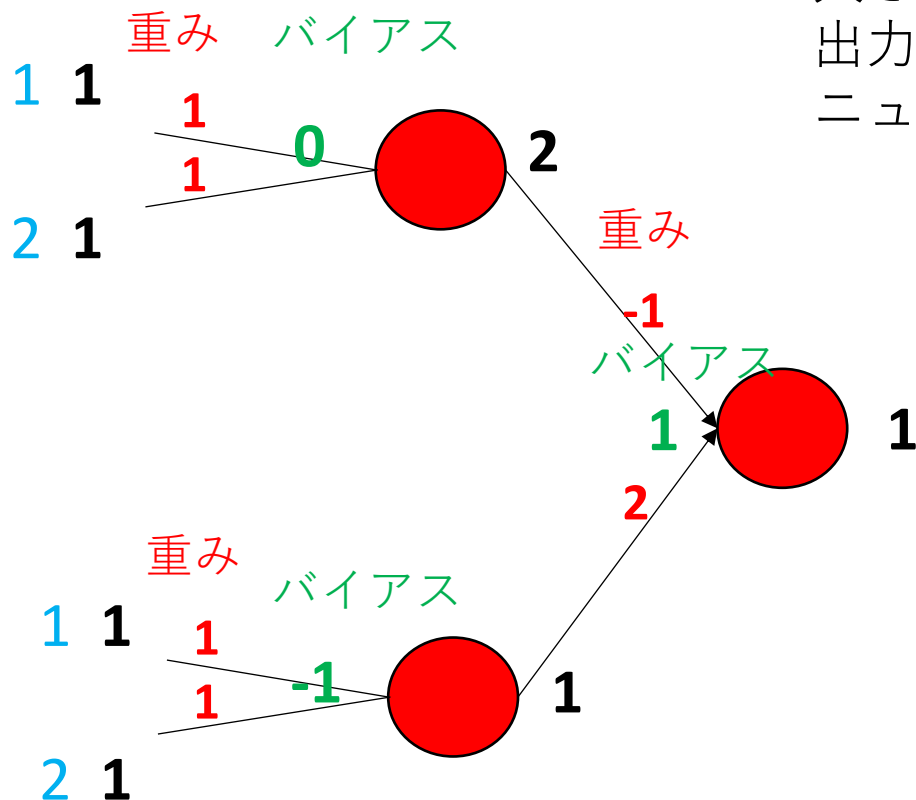
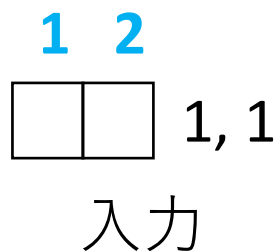
大きなプラスの値を
出力している
ニューロン = 活性化



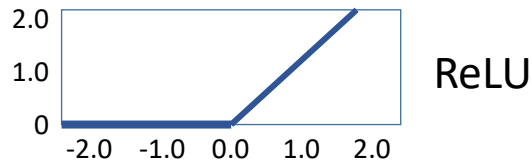
3つのニューロンの活性化関数はすべて ReLU



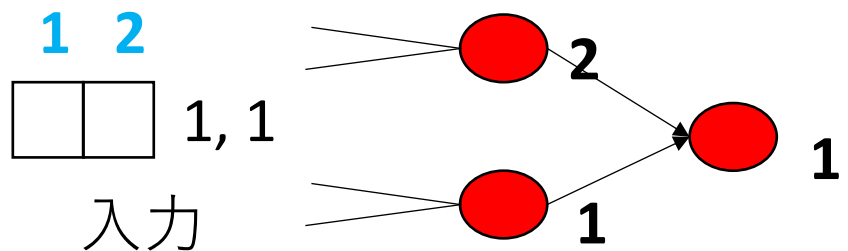
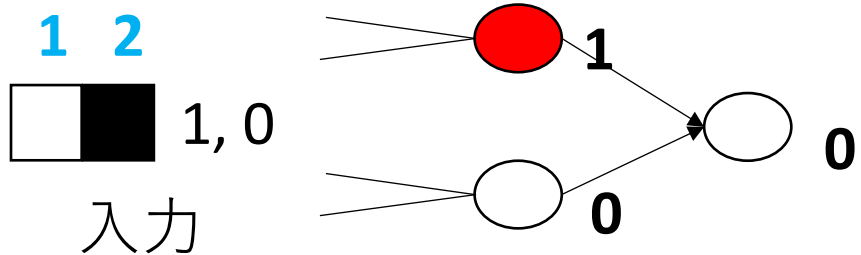
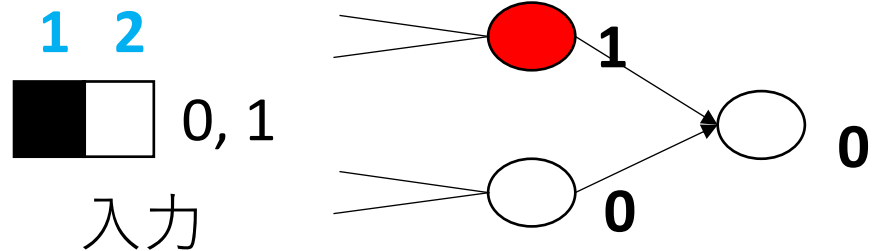
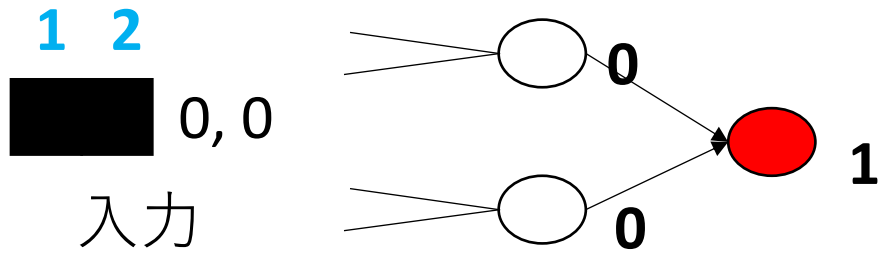
大きなプラスの値を
出力している
ニューロン = 活性化



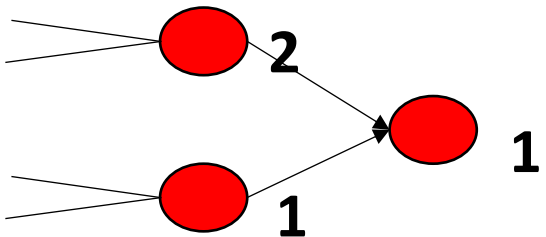
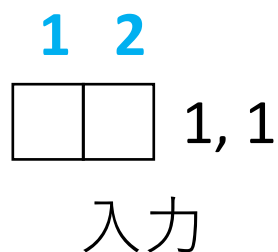
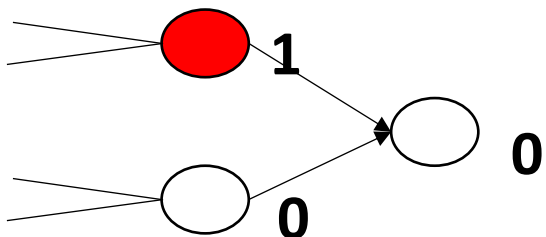
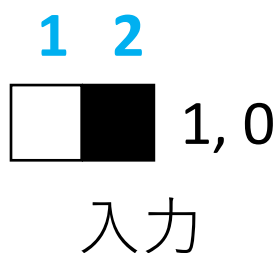
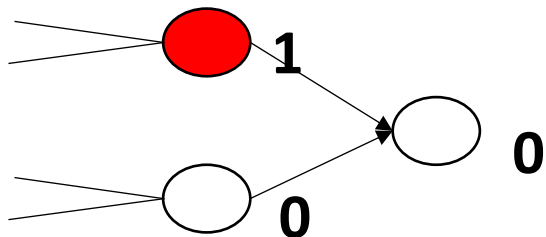
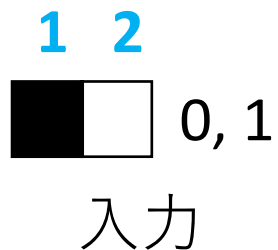
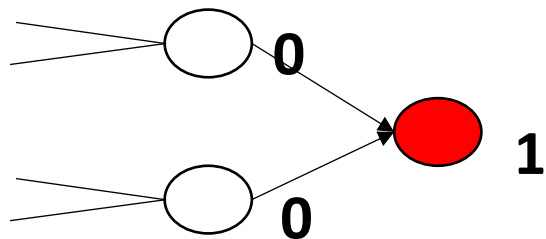
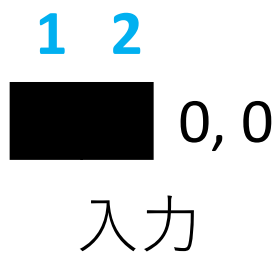
3つのニューロンの活性化関数はすべて ReLU



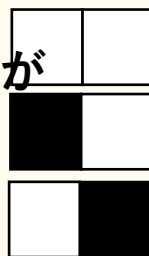
入力の変化による活性度の変化



ニューロンは特定のパターンに応じて活性化する



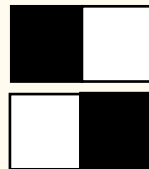
ニューロンが
識別する
パターン



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

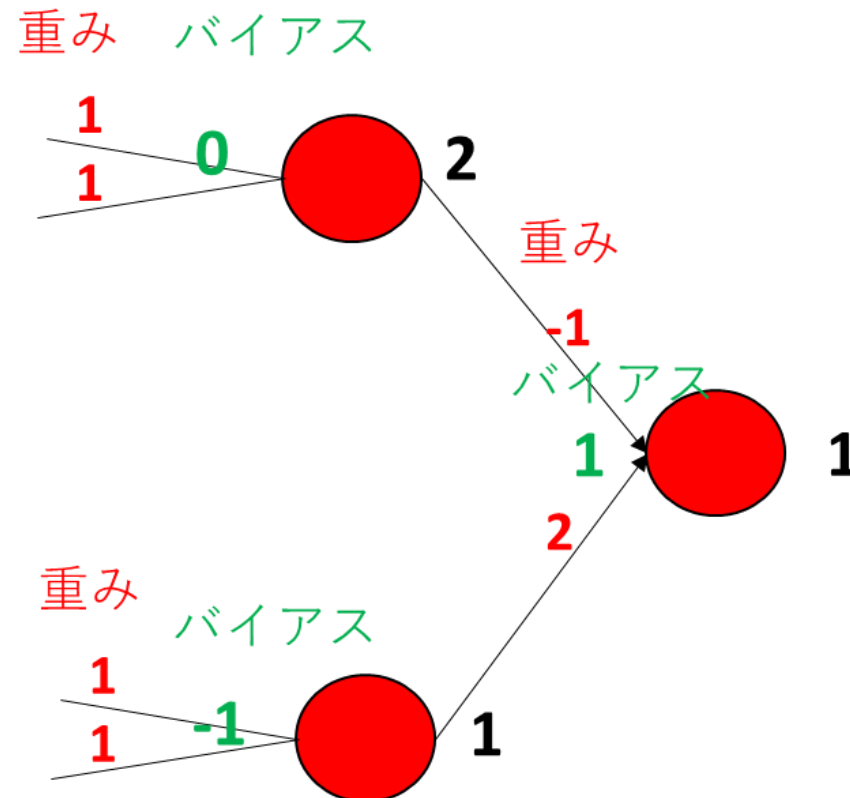


演習 ニューラルネット ワーク

演習の狙い



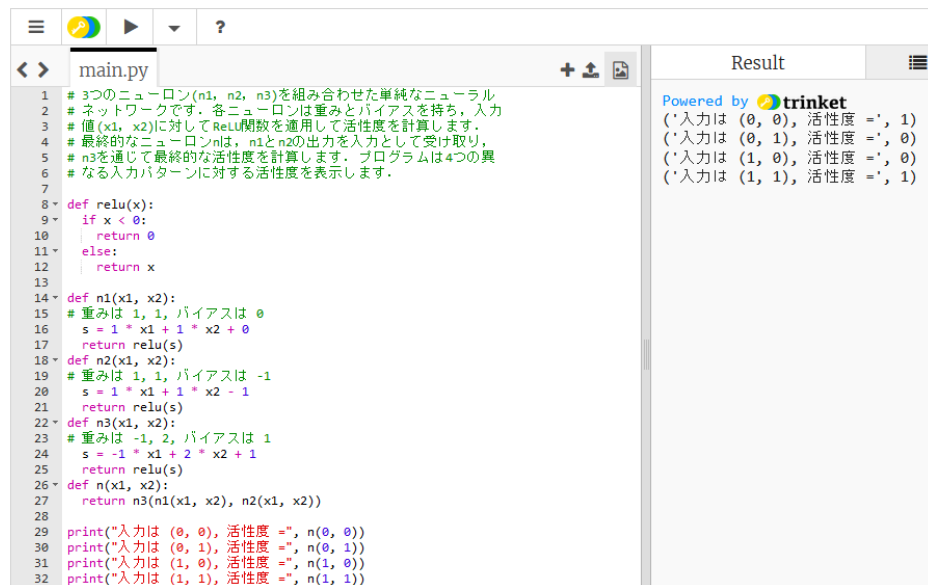
- 複数のニューロンが層を形成
- 1層の場合より複雑な判断が可能になる
- ニューラルネットワークによる高度な情報処理の基礎を体験



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



```
1 # 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
2 # 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
3 # 値(x1, x2)に対してReLU関数を用いて活性化度を計算します。
4 # 最終的なニューロンn3は、n1とn2の出力を入力として受け取り、
5 # n3を通じて最終的な活性化度を計算します。プログラムは4つの異
6 # なる入力パターンに対する活性化度を表示します。
7
8 def relu(x):
9     if x < 0:
10         return 0
11     else:
12         return x
13
14 def n1(x1, x2):
15     # 重みは 1, 1, バイアスは 0
16     s = 1 * x1 + 1 * x2 + 0
17     return relu(s)
18
19 def n2(x1, x2):
20     # 重みは 1, 1, バイアスは -1
21     s = 1 * x1 + 1 * x2 - 1
22     return relu(s)
23
24 def n3(x1, x2):
25     # 重みは -1, 2, バイアスは 1
26     s = -1 * x1 + 2 * x2 + 1
27     return relu(s)
28
29 def n(x1, x2):
30     return n3(n1(x1, x2), n2(x1, x2))
31
32 print("入力は (0, 0), 活性化度 =", n(0, 0))
33 print("入力は (0, 1), 活性化度 =", n(0, 1))
34 print("入力は (1, 0), 活性化度 =", n(1, 0))
35 print("入力は (1, 1), 活性化度 =", n(1, 1))
```

Result

Powered by  trinket

```
('入力は (0, 0), 活性化度 =' , 1)
('入力は (0, 1), 活性化度 =' , 0)
('入力は (1, 0), 活性化度 =' , 0)
('入力は (1, 1), 活性化度 =' , 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

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。



5-4. ディープラーニング の特徴と応用

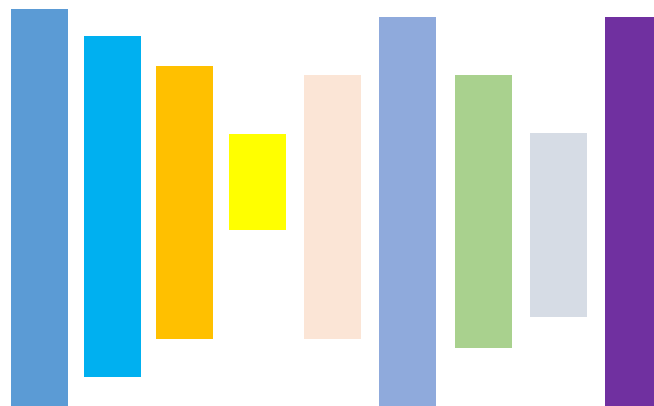
ディープニューラルネットワーク



- ディープニューラルネットワークは、
層が深い（層の数が多い）ニューラルネットワーク



層の数が少ない（浅い）

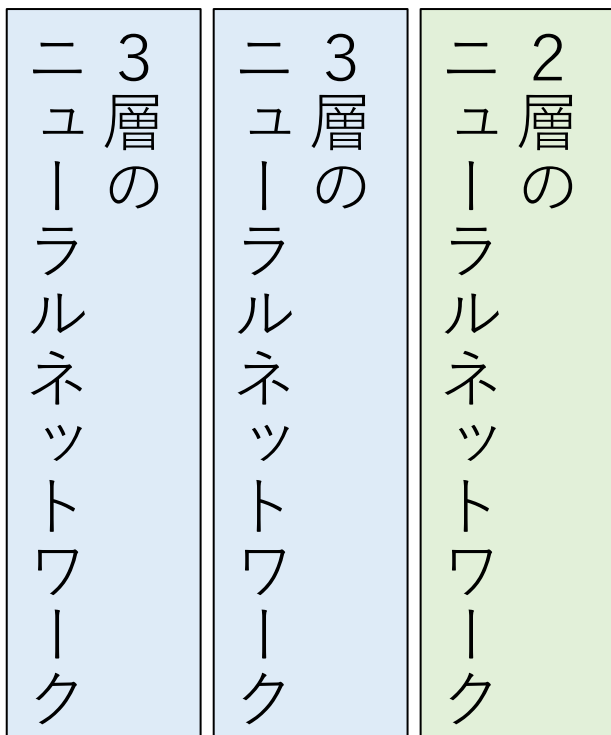


層の数が多い（深い）

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



- **多様なデータに対応**

例：画像、テキスト、音声、動画など

- **応用の広さ**

例：画像認識、自然言語処理、音声認識など

- **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。

- **自動でのタスク実行**

パターン抽出ならびにタスク実行は自動で行われる。

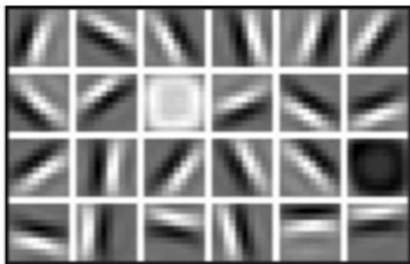
さまざまなレベルのパターン

Why Deep Learning?

Hand engineered features are time consuming, brittle, and not scalable in practice

Can we learn the **underlying features** directly from data?

Low Level Features



Lines & Edges

Mid Level Features



Eyes & Nose & Ears

High Level Features



Facial Structure

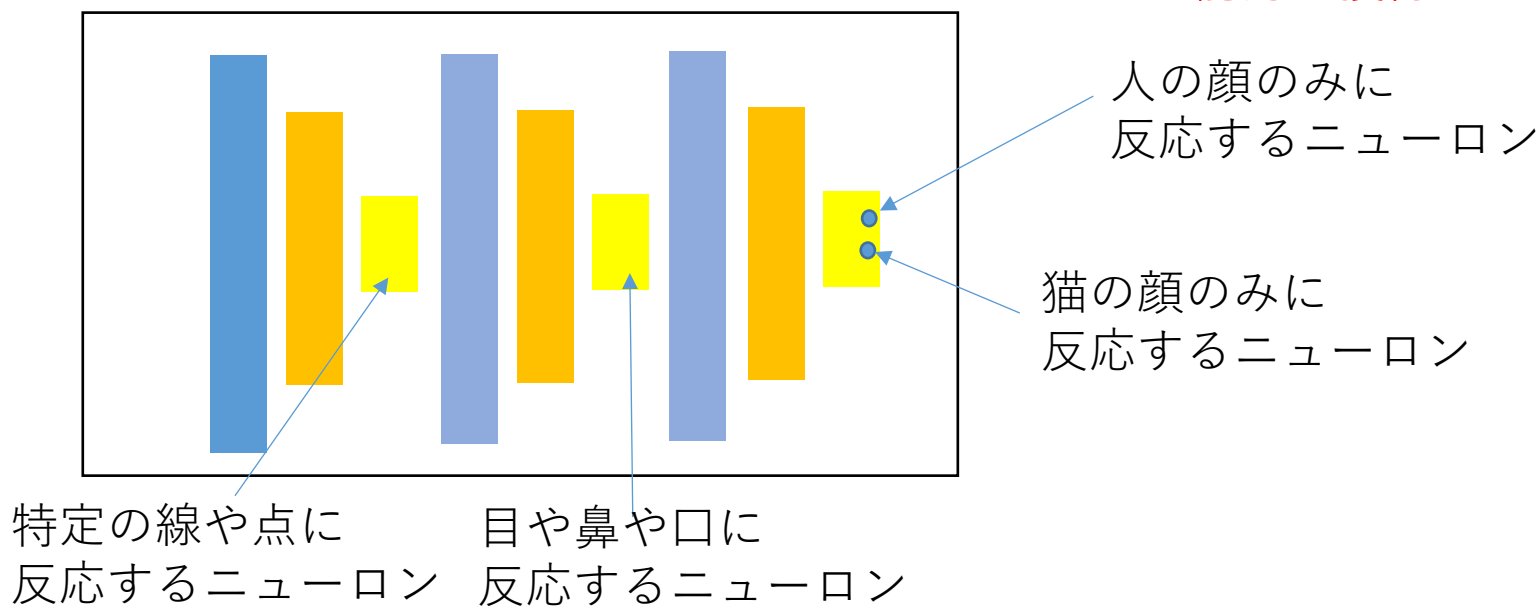


線や点のレベル 目, 鼻, 耳のレベル 顔の構造のレベル

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



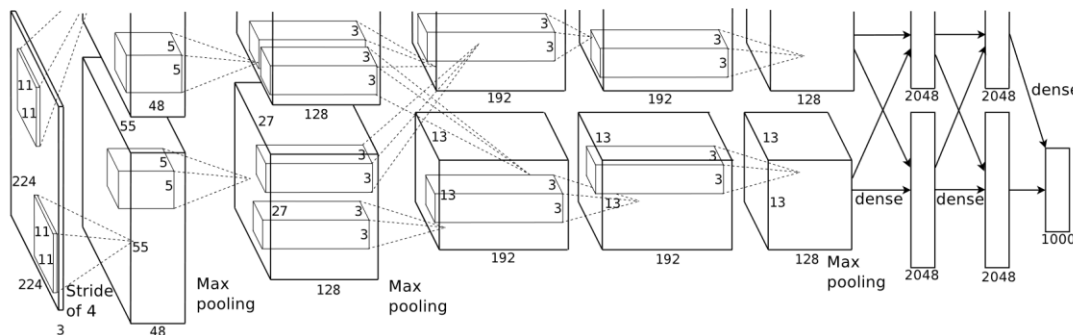
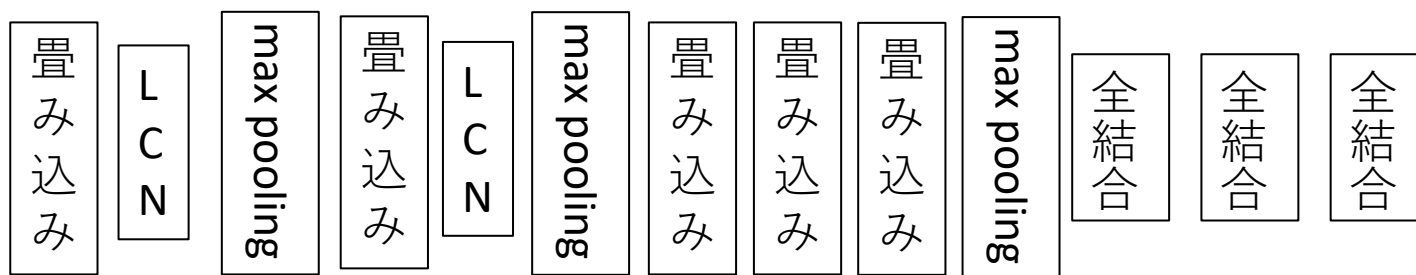
文献: Building high-level features using large scale unsupervised learning

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

Why Deep Learning?

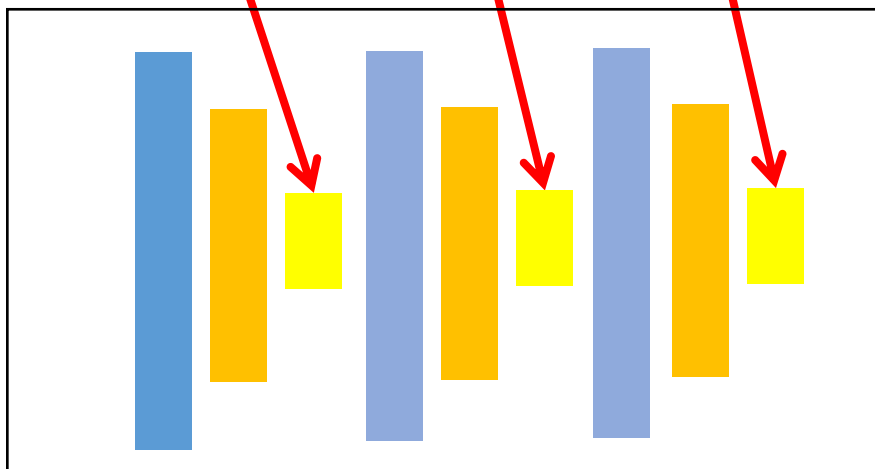
Hand engineered features are time consuming, brittle, and not scalable in practice

Can we learn the **underlying features** directly from data?

Low Level Features	Mid Level Features	High Level Features
Lines & Edges	Eyes & Nose & Ears	Facial Structure

Massachusetts Institute of Technology
6.S191 Introduction to Deep Learning
introtodeeplearning.com @MITDeepLearning
1/18/21

IntroToDeepLearning.com



「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



- **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



GT: horse cart
1: horse cart
2: minibus
3: oxcart
4: stretcher
5: half track



GT: birdhouse
1: birdhouse
2: sliding door
3: window screen
4: mailbox
5: pot



GT: forklift
1: forklift
2: garbage truck
3: tow truck
4: trailer truck
5: go-kart



GT: coucal
1: coucal
2: indigo bunting
3: lorikeet
4: walking stick
5: custard apple



GT: komondor
1: komondor
2: patio
3: llama
4: mobile home
5: Old English sheepdog



GT: yellow lady's slipper
1: yellow lady's slipper
2: slug
3: hen-of-the-woods
4: stinkhorn
5: coral fungus

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)

**ImageNet データセット
の画像分類の結果**

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.

多層でも学習が**成功するため**のさまざまな工夫

- 活性化関数: ReLU など
- ドロップアウト
- He の初期化
- 大量の**データ**の準備
- 繰り返し学習による**学習不足**の解消
- データ拡張による**データ**の増量
- **ニューラルネットワーク**を**高速にシミュレーション**できる高性能のコンピュータ

ディープラーニングの応用と特徴

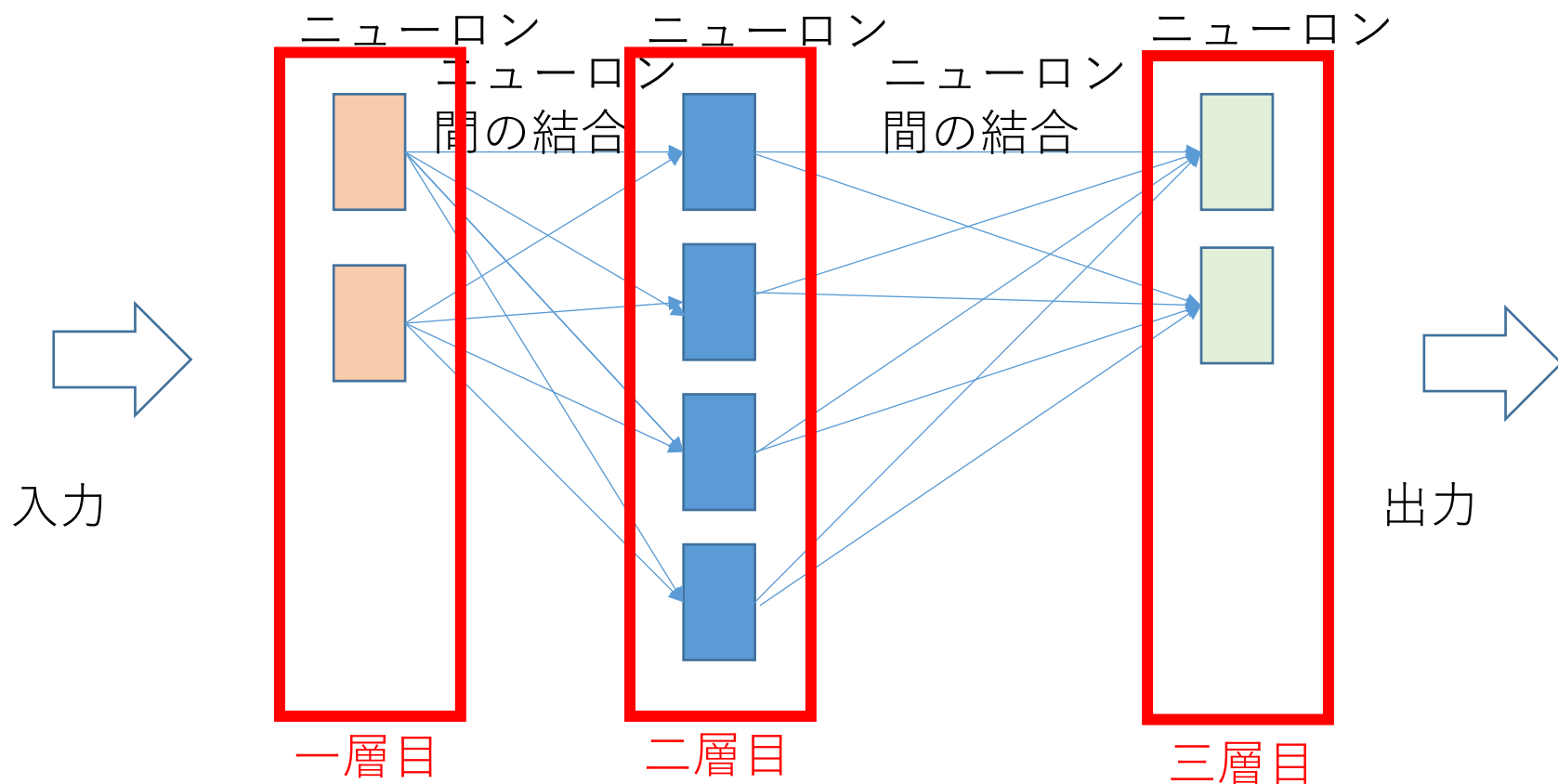


- **ディープラーニング**は**機械学習**の一種であり、人工**ニューラルネットワーク**を使用して**データから学習**し、複雑な**タスクを実行**する技術
- 「ディープ」の名前は、**多層のニューラルネットワーク**を使用することに由来
- ディープラーニングが広く利用される理由は、**多様なデータに適用**でき、**さまざまなタスク**で高性能を発揮するため。例えば、**画像認識**、**自然言語処理**、**音声認識**など。



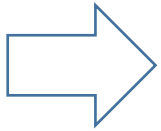
5-5. ニューラルネットワークを用いた分類

層構造とデータの流れ

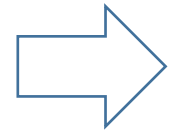
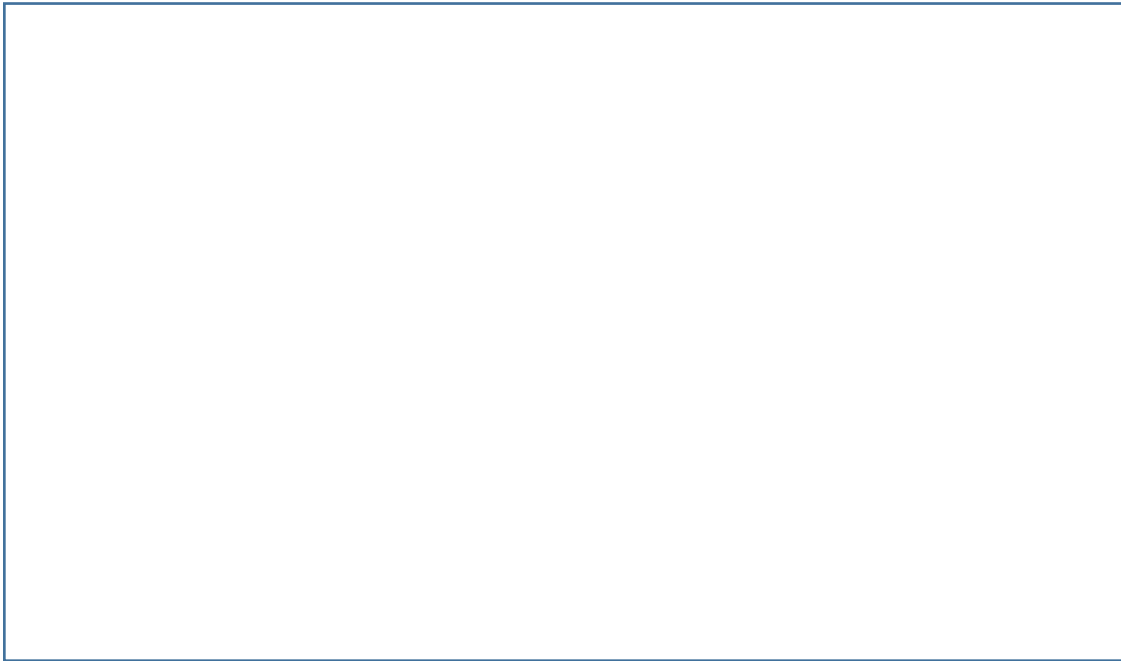


- 層構造では、データは、入力から出力への一方向に流れる
- 各層は、同じ種類のニューロンで構成

3 種類の中から 1 つに分類する場合



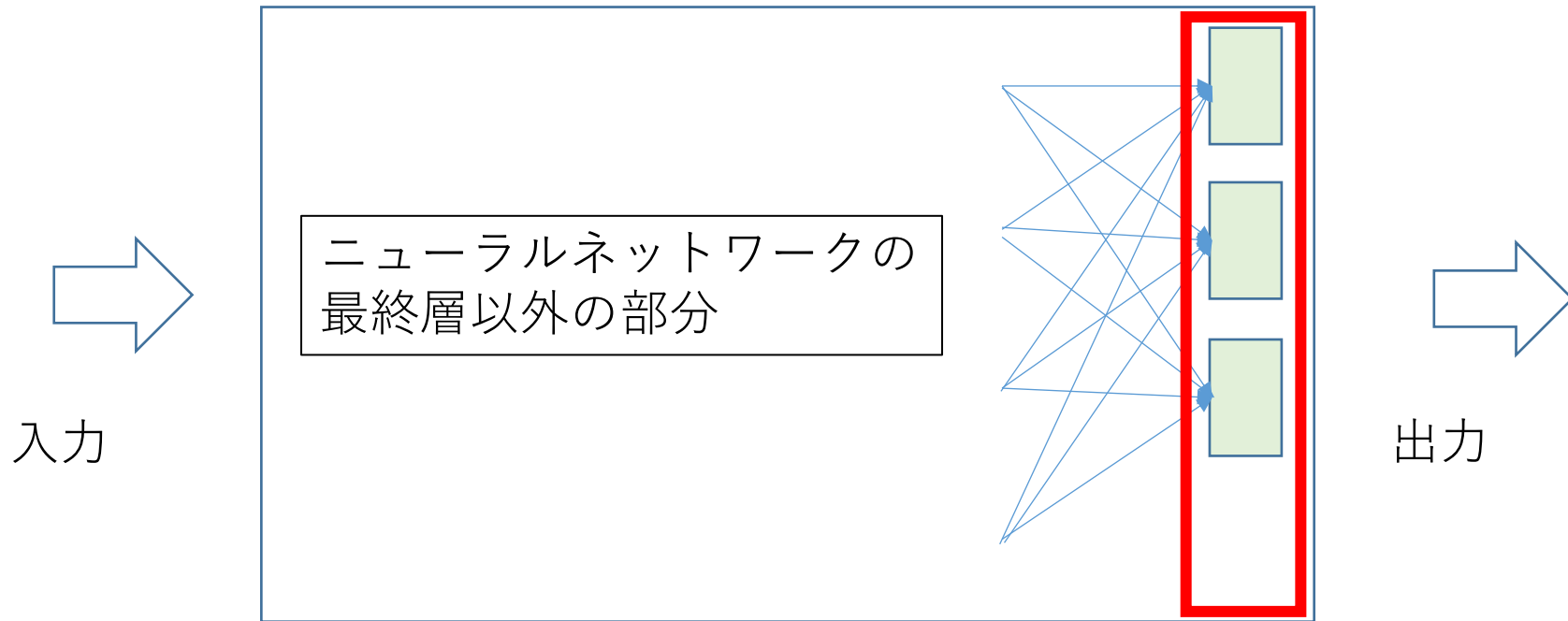
入力



分類結果
0 または 1
または 2

3 種類の中から 1 つに分類する場合

最終層のニューロン数：3 にする ニューロン

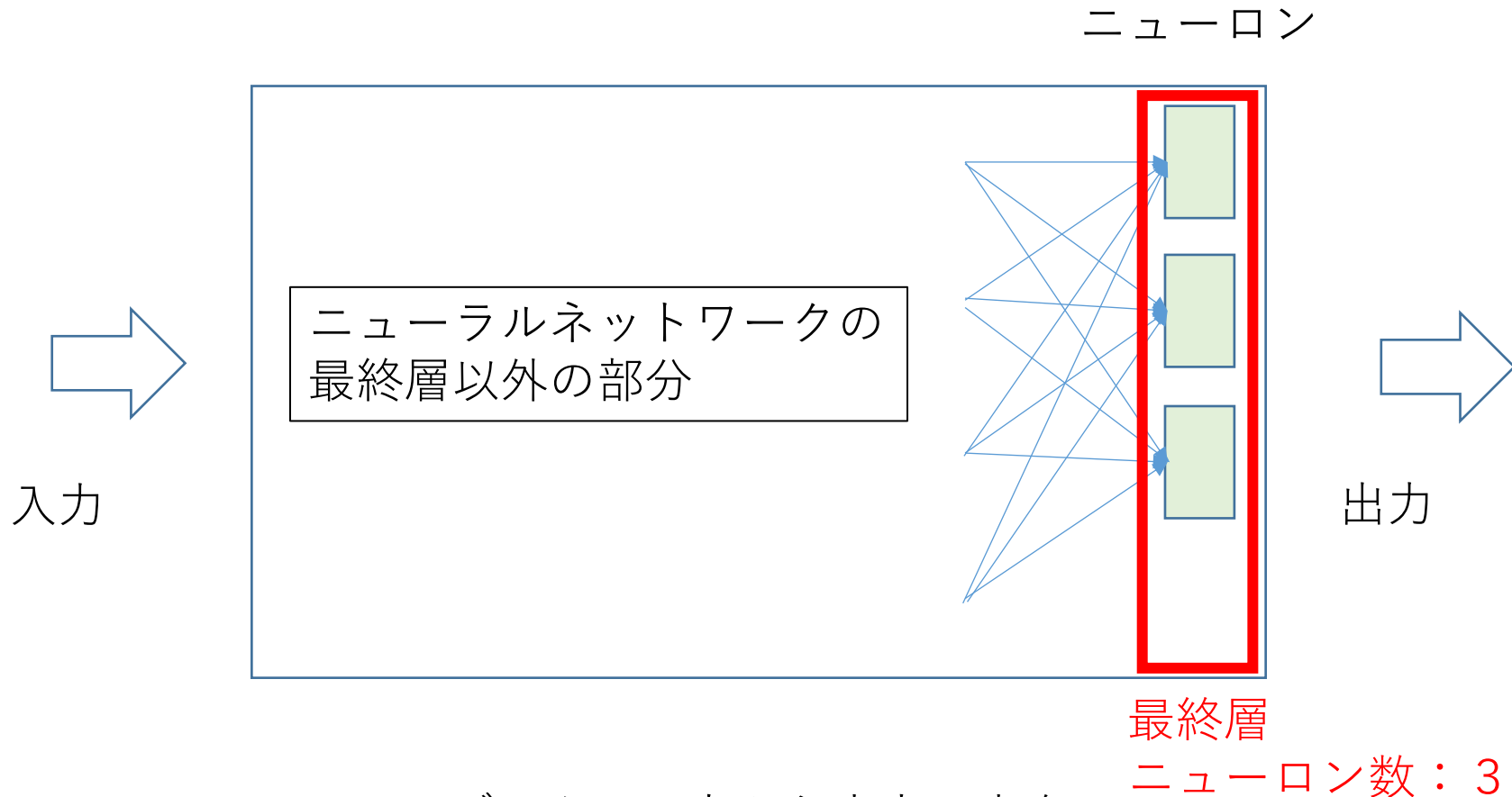


最終層

ニューロン数：3

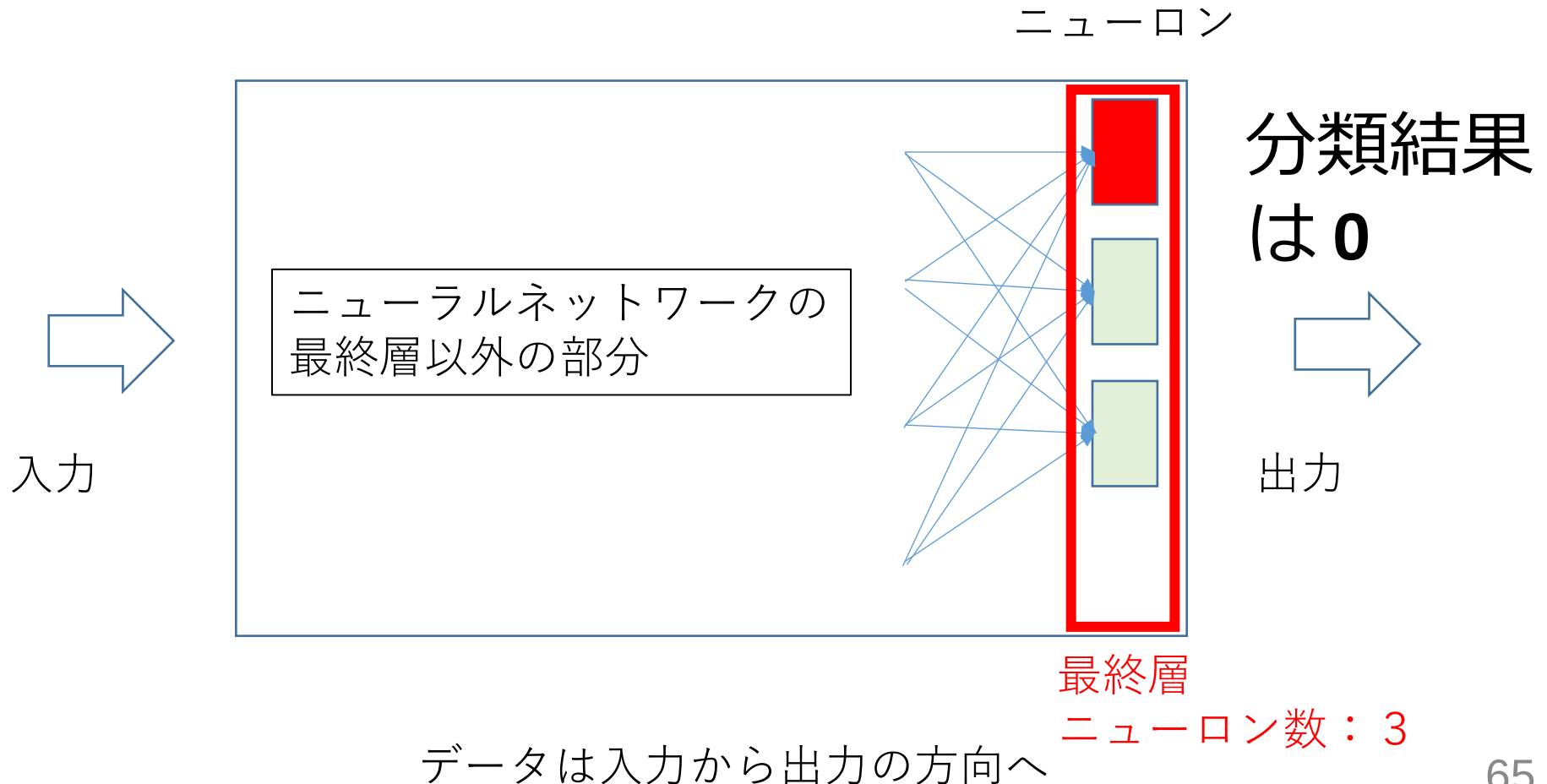
データは入力から出力の方向へ

最終層について、1つが強く 活性化するように調整

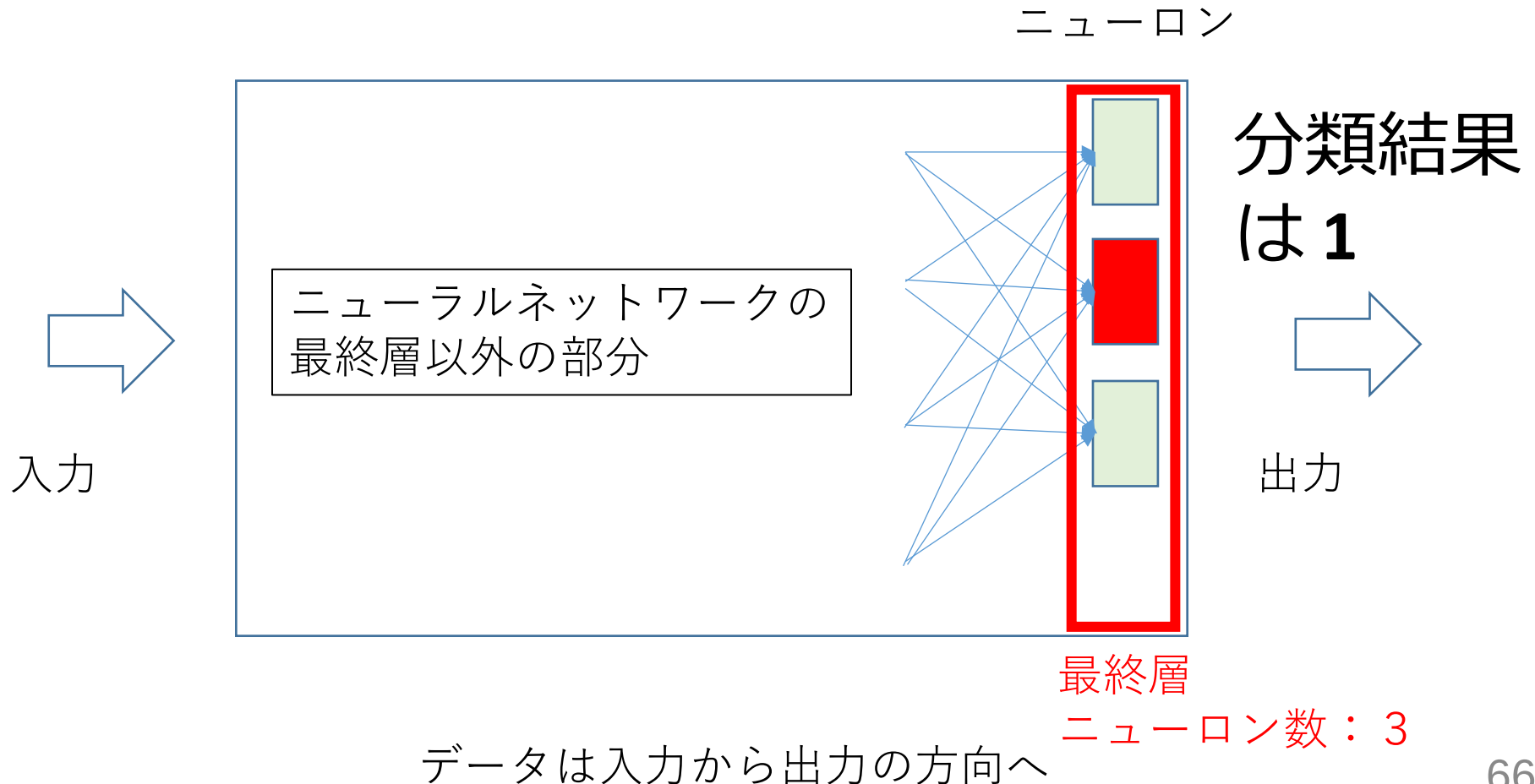


データは入力から出力の方向へ

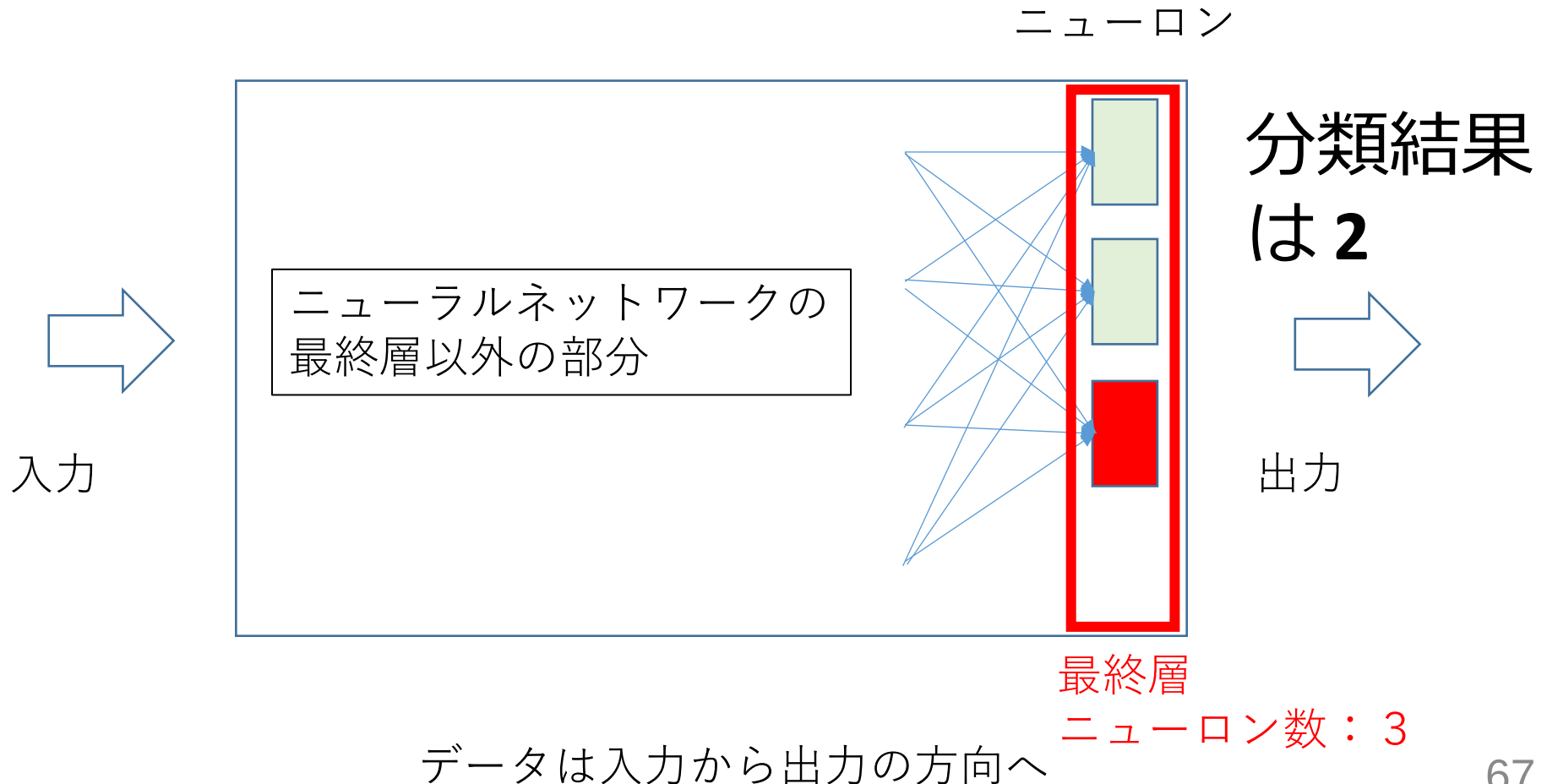
最終層について、1つが強く 活性化するように調整



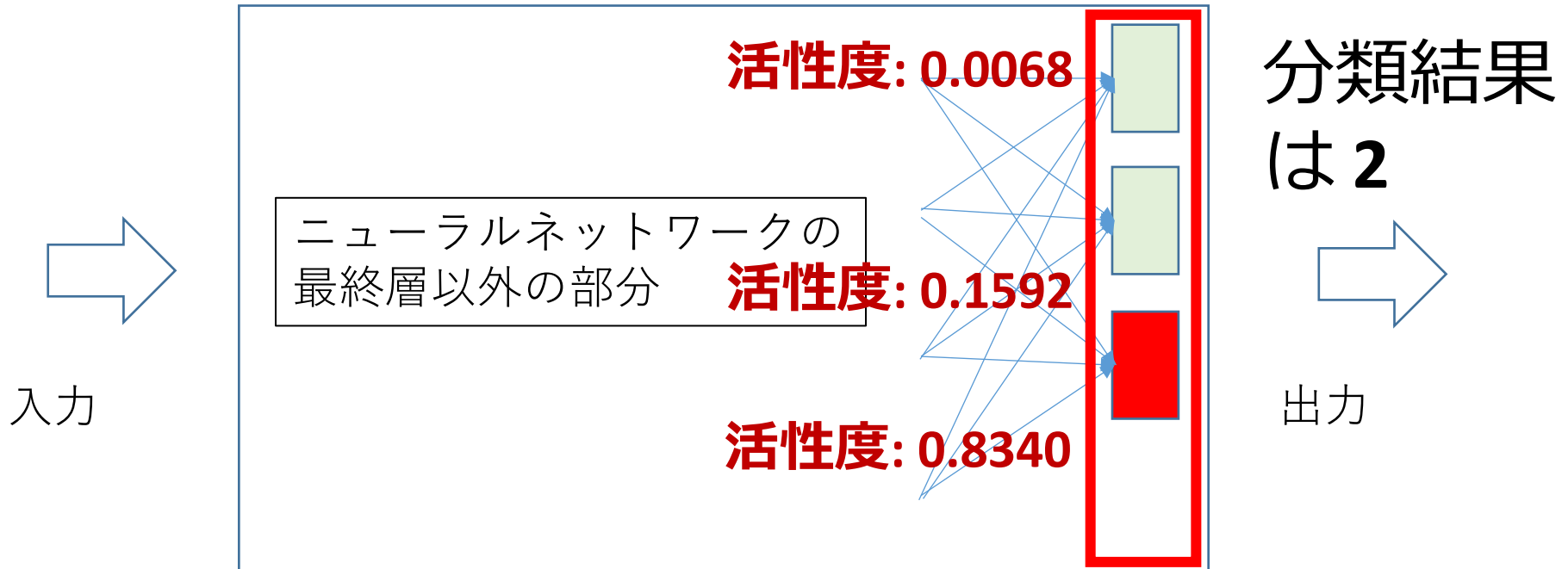
最終層について、1つが強く 活性化するように調整



最終層について、1つが強く 活性化するように調整



最終層について、1つが強く
活性化するように調整

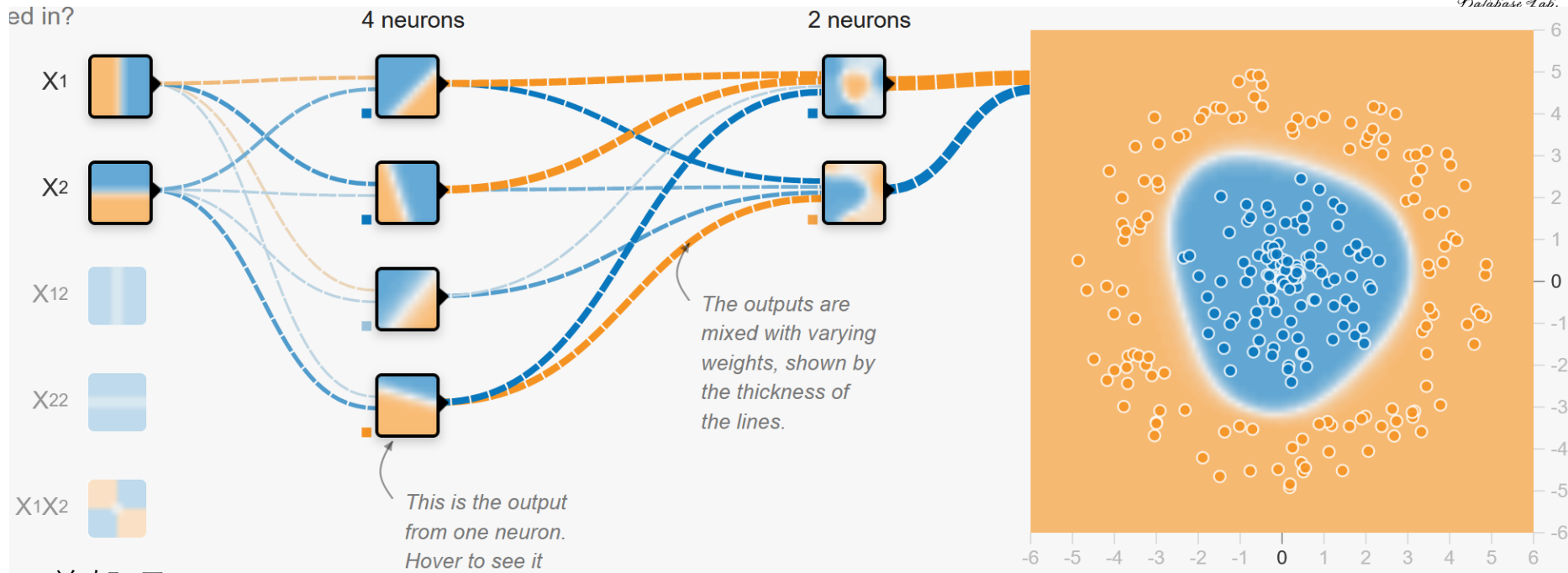


実際には、**活性化度**は 0 から 1 のような数値である。
最も**活性化度**の値が高いものが選ばれて、分類結果となる

ニューラルネットワークを用いた分類

ニューラルネットワークを分類に使うとき

- 最終層のニューロンで、最も活性度の値の高いものが選ばれて、分類結果となる
- 誤差がある



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

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。

- ニューラルネットワークの仕組みを基礎から学び、人工知能（AI）の技術への理解の深まり
- 工学を学ぶことのへの意欲の高まり
- ディープラーニング（深層学習）の実践的活用に役立つ知識とスキルの向上
- 人工知能（AI）の応用可能性の広さを認識し、様々な分野での課題解決に挑戦する意欲が湧く。

図解と例示を通じて学び、学習意欲を高めましょう。技術の面白さを実感しましょう。