

初めてディープラーニング学ぶ人の ための数学入門

--- ニューラルネットで行列を理解する



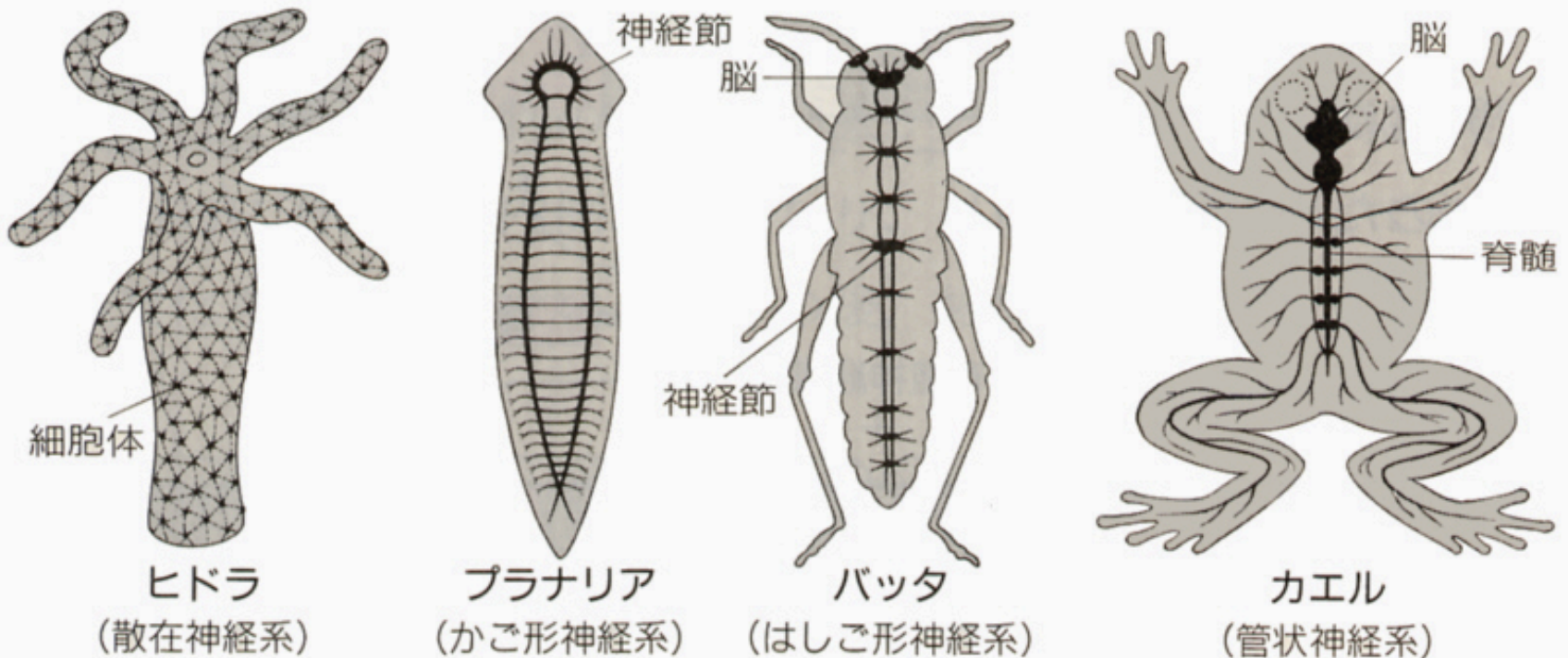
Part I

Deep Neural Network



モデルとしての生物のニューラル・ネットワーク

クラゲだって神経はある



<http://www.tmd.ac.jp/artsci/biol/textlife/neuron.htm>

「神経系の起源と進化」

<http://bit.ly/1qR2Dmq>

NEURONS AND SYNAPSES
This wiring diagram shows more than 4,500 of the 8,000 neuron-to-neuron connections, or synapses, in the worm's nervous system. Each dot represents a single neuron:

- **SENSORY NEURONS**
detect external stimuli
- **CONNECTOR NEURONS**
relay signals
- **MOTOR NEURONS**
control muscle movement

線虫のC. Eleganceは、すべての神経の接続がわかっている唯一の生物である。
302の神経と8,000のシナプスがある。

SOCIAL AND SOLITARY WORMS
Most worms in nature congregate in clumps, a behavior controlled by two neurons called RMG, one of which is highlighted at left.

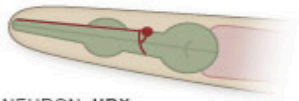
RMG combines information from several sensory neurons, also highlighted, and then signals the worm's muscles to move toward nearby worms if conditions are right.

But if a specific gene in RMG is activated, as it is in many laboratory worms, the worm will switch to more solitary behavior.

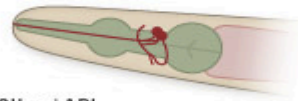
1987年

<http://goo.gl/0lbzRg>

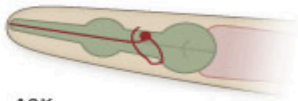
FOLLOWING NEURONS
Six neurons highlighted in the diagram are shown below. Most neurons are located in the head, though some, like AVA, run the length of the worm's body.



NEURON: URX
Senses oxygen in the environment



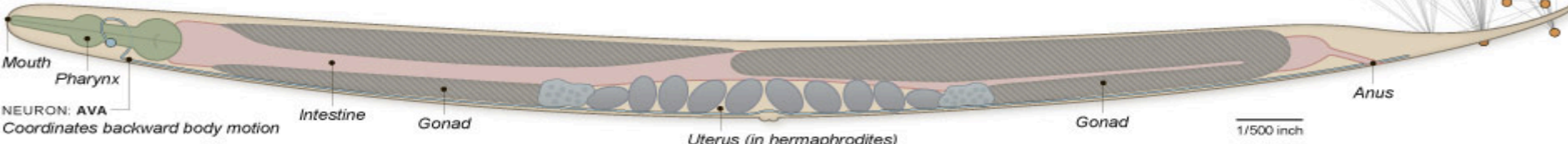
ASH and ADL
Sense poisons and other noxious stimuli



ASK
Senses food signals and detects sex pheromones from other worms



RMG
Hub of a network of neurons that controls social behavior



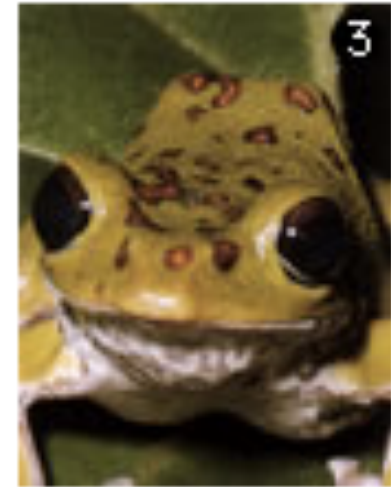
アノマロカリス



「視覚」の誕生と発展

- 生物の進化の歴史の中で、「視覚」の能力は、5億年前の「カンブリア紀」に獲得された。それは、長い長い歴史を持っている。当時の食物連鎖の頂点に立っていた「アノマロカリス」は、立派な眼を持っていた。
- 「眼」の獲得、同じことだが、「視覚」の能力の獲得は、「捕食者」にとっても（獲物を見つける）、「被捕食者」にとっても（捕食者から逃れる）、その「生き残り」に、死活的に重要な意味を持つことになった。人間を含めて、すべての動物で、強い淘汰の圧力のもとで、視覚処理の能力は、高度に発達している。
- 同時に、この「視覚」の能力は、追いかける、捉える、あるいは、逃げるといった「運動能力」に深く結びついている

多くの生物は目を持っている



—— ゲーリングによる「目の遺伝子」Pax-6の発見 ——

https://www.brh.co.jp/seimeishi/journal/012/ss_1.html

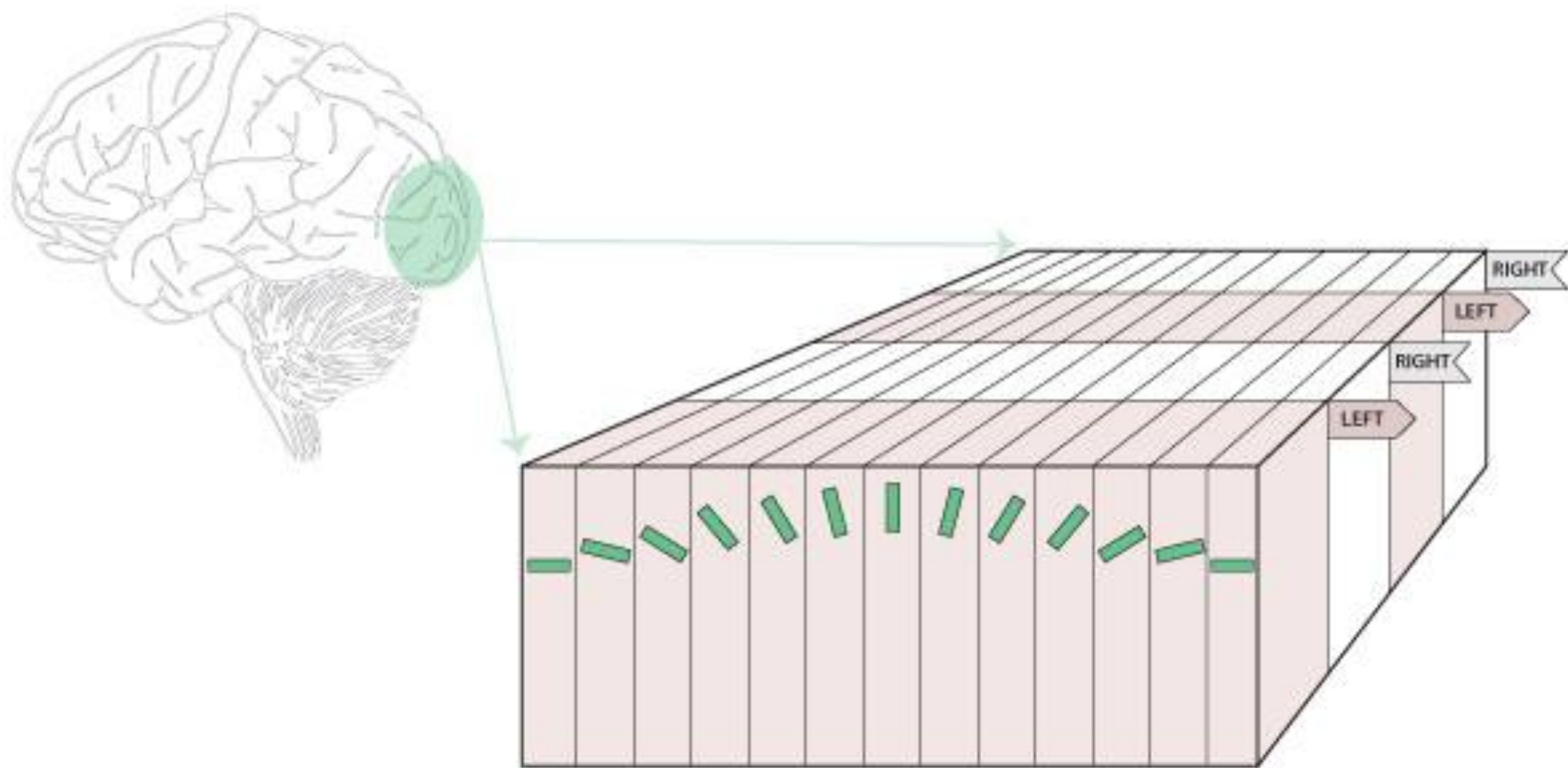


視覚と運動能力は、強く結びついている



センサーを持つマシンの自律的運動
現在のディープラーニングが得意な分野

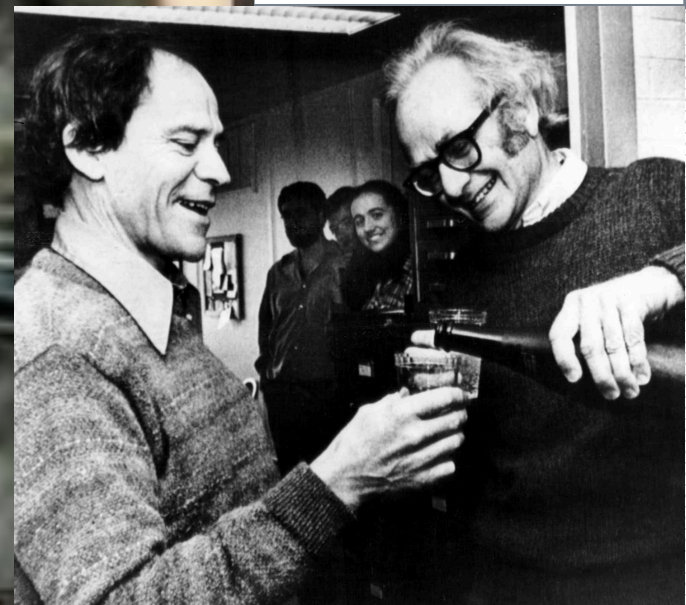
60年代から70年代にかけての、Hubel と Wieselの
大脳視覚野の研究は、各方面に大きな影響を与えた。
ニューラル・ネットワークの最初の研究も、こうした影響の
もとで始まった。



1960年代

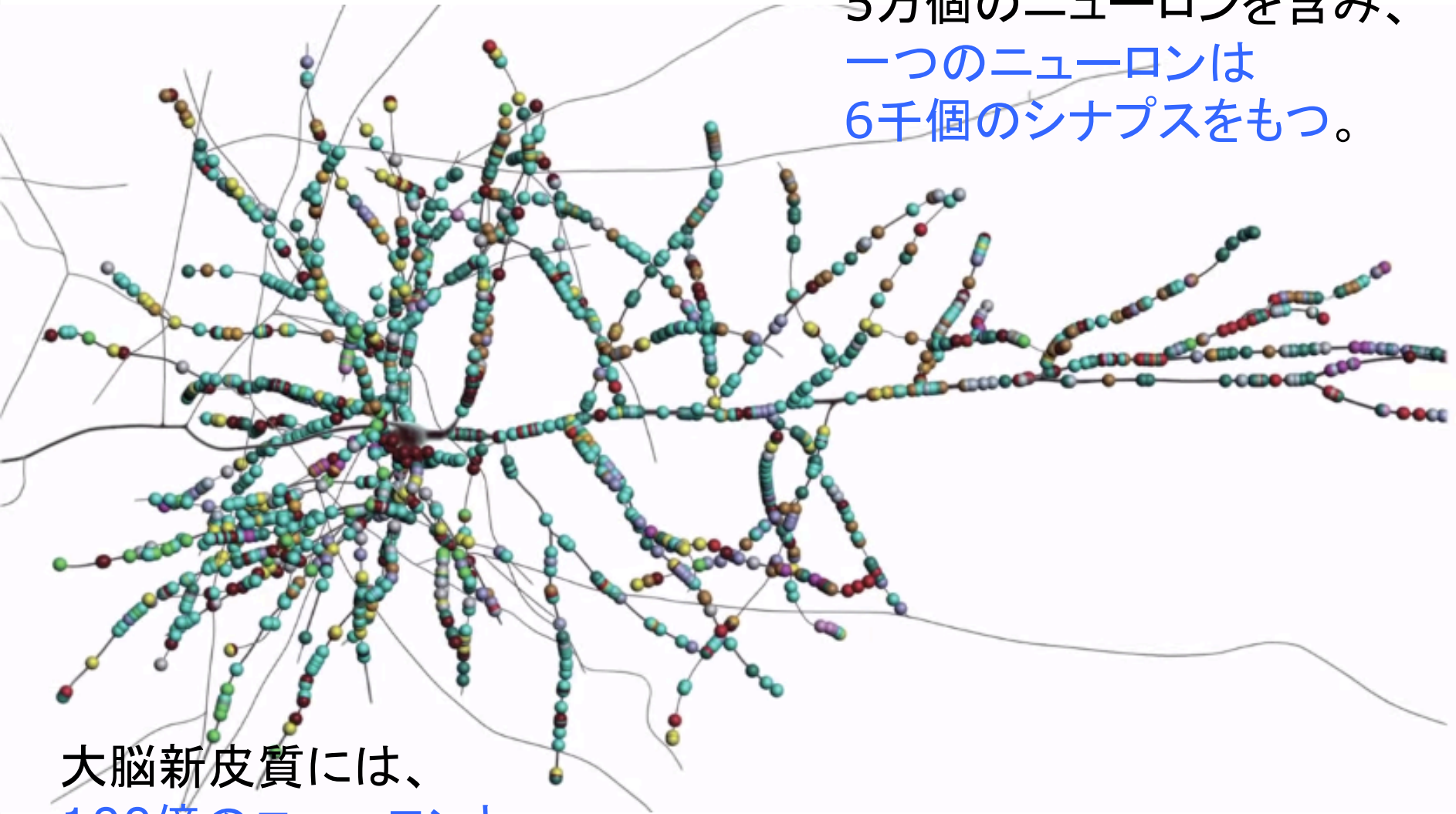
David H. Hubel

Torsten Wiesel





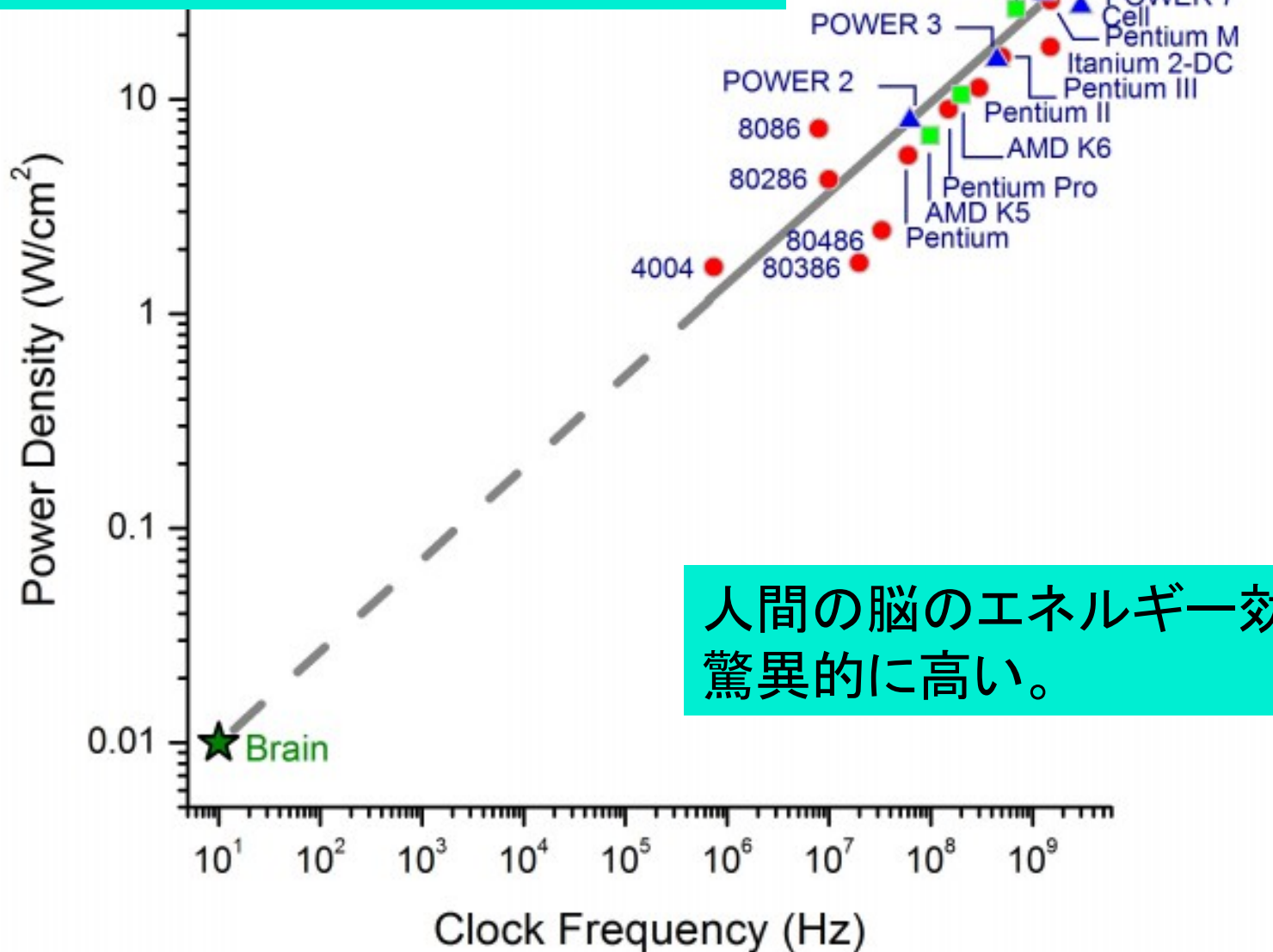
脳は、1mm立方に
5万個のニューロンを含み、
一つのニューロンは
6千個のシナプスをもつ。



大脳新皮質には、
100億のニューロンと
60兆個のシナプスがある！

AIのハードの世界でも大きな変化が進んでいるのだが、留意すべきこと

....



人間の脳のエネルギー効率は、驚異的に高い。



「人間と機械」という対比だけでなく、「生物と機械」という対比でも、考えることは面白い。

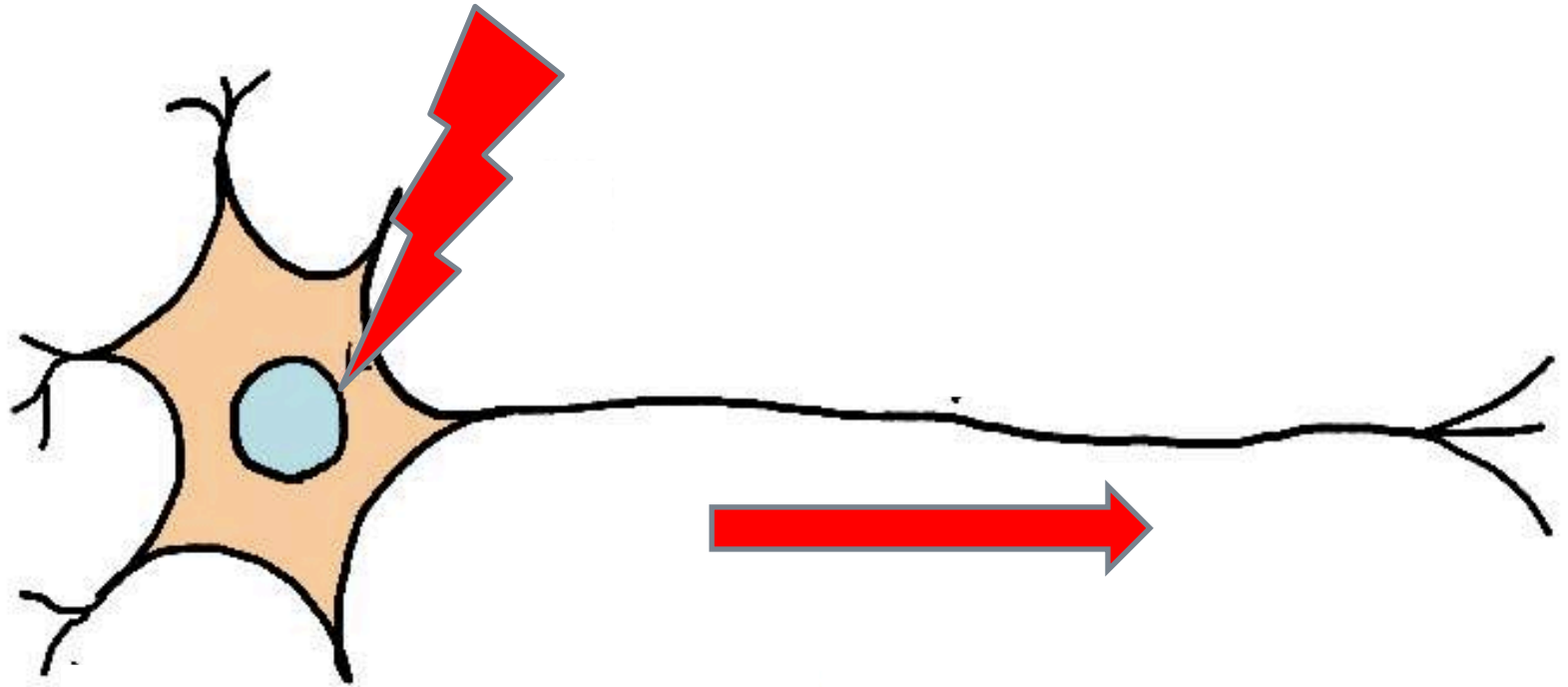
ニューラル・ネットワークが、今後、大活躍するであろう「感覚-運動系」でも、生物の能力は、驚異的である。



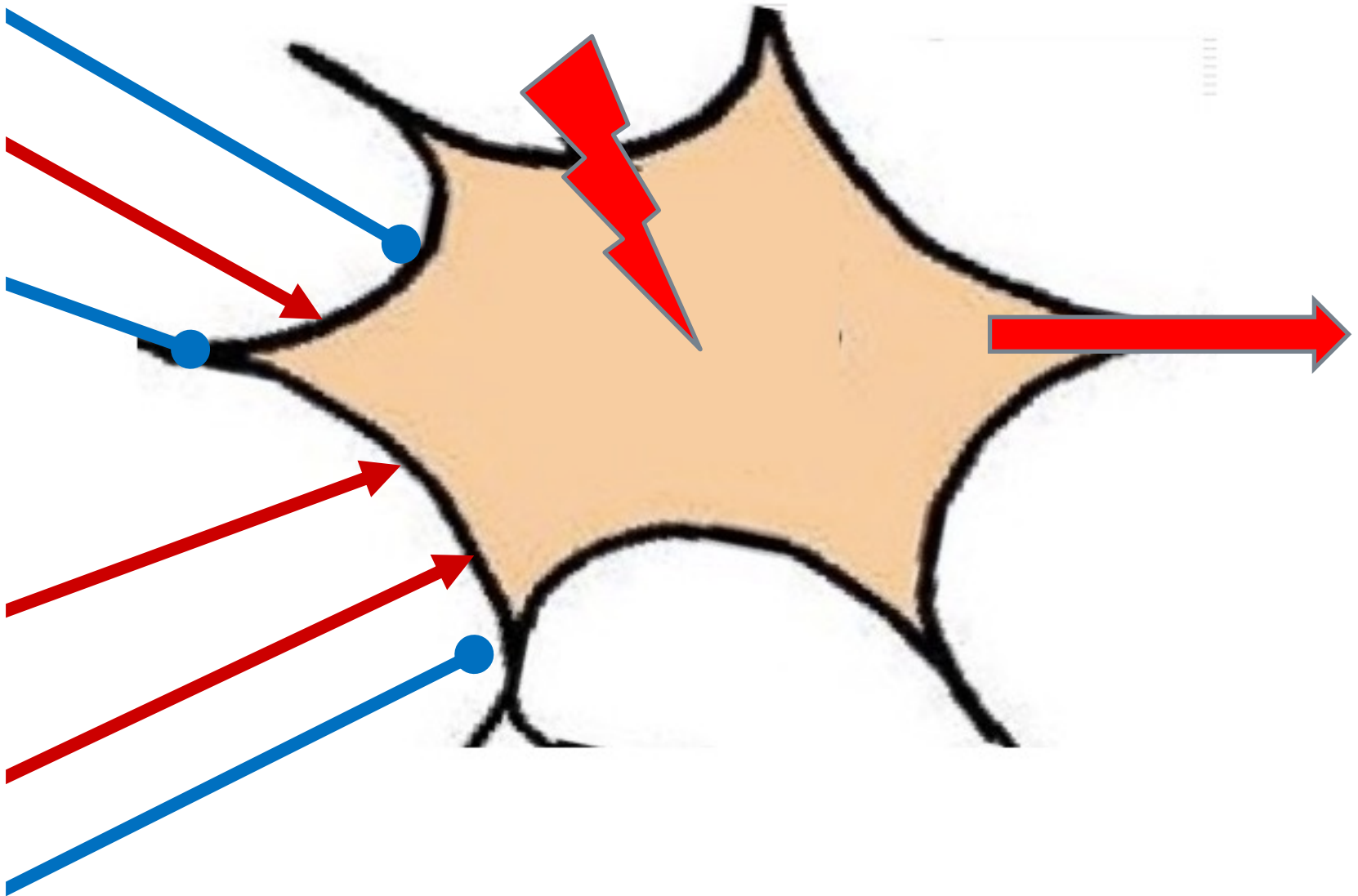
一つのニューロンの動きを考える

以下の文中に現れる「ニューロン」は、コンピュータ上でシミュレートされた偽の「ニューロン」である。ただし、基本的な動作原理は、本物のニューロンを真似ていることを知っておくことは、役に立つと思う。

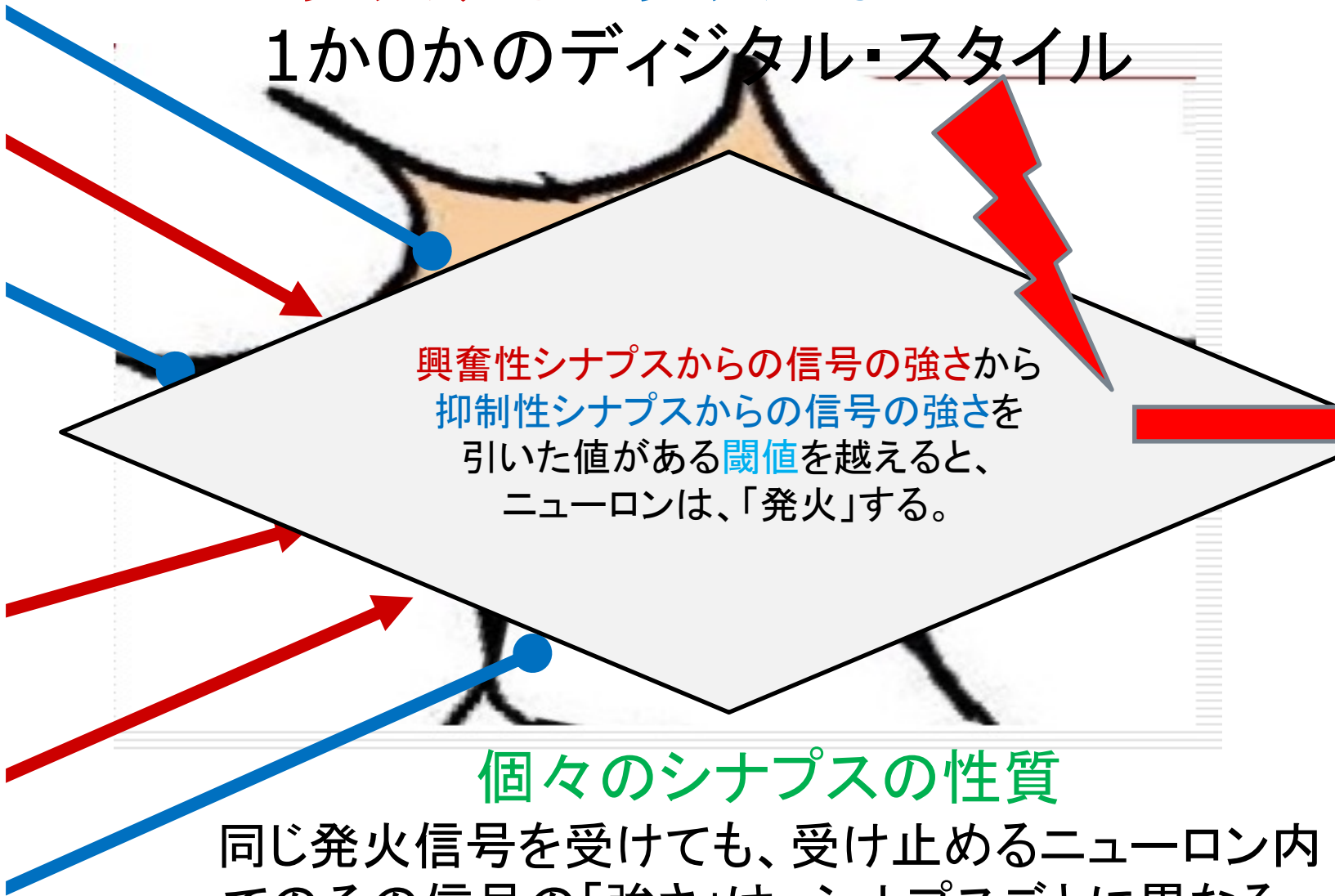
ニューロンは、
いつ発火するのか？



ニューロンは、
いつ発火するのか？



ニューロンの働きは、
発火するか発火しないかの
1か0かのデジタル・スタイル



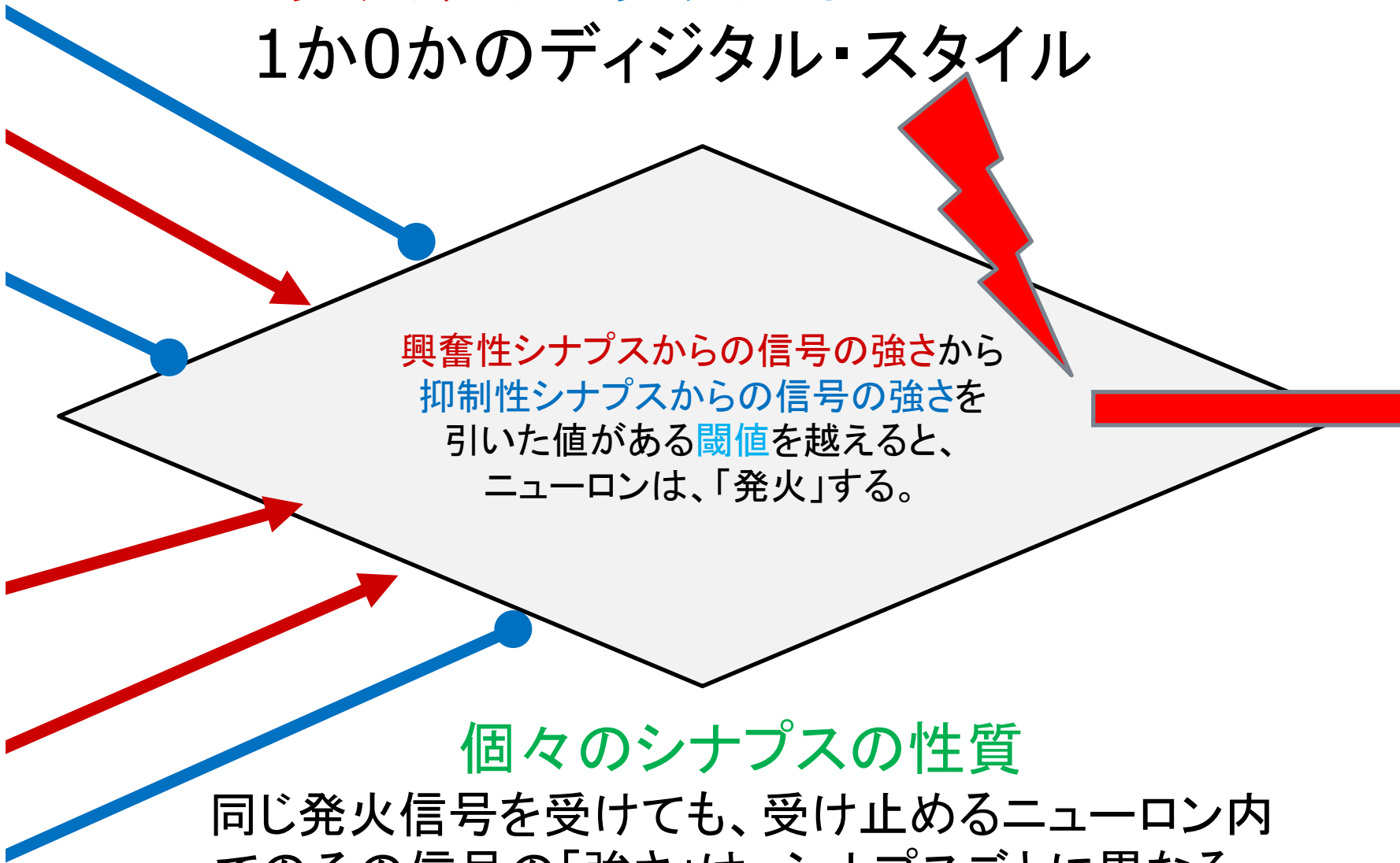
興奮性シナプスからの信号の強さから
抑制性シナプスからの信号の強さを
引いた値がある閾値を越えると、
ニューロンは、「発火」する。

The diagram shows a central gray diamond-shaped neuron. Several arrows point towards it from the left: two blue arrows (excitatory) and two red arrows (inhibitory). One blue arrow has a blue dot at its tip, and one red arrow has a red dot. A large red lightning bolt points towards the right side of the neuron. A red horizontal bar is on the right edge of the neuron. The background is a faint, stylized drawing of a person's head and shoulders.

個々のシナプスの性質

同じ発火信号を受けても、受け止めるニューロン内
でのその信号の「強さ」は、シナプスごとに異なる。

ニューロンの働きは、
発火するか発火しないかの
1か0かのデジタル・スタイル



The diagram illustrates a neuron model. A central gray diamond represents the neuron's cell body. Several colored lines (blue and red) with arrows point towards the diamond, representing synaptic inputs. Some of these lines have small blue circles at their points of contact with the diamond. A large red lightning bolt symbol points towards the right side of the diamond, representing an action potential. A thick red horizontal bar is positioned to the right of the diamond, representing the output of the neuron. Inside the diamond, there is text explaining the firing process. Below the diamond, there is text explaining the properties of individual synapses.

興奮性シナプスからの信号の強さから
抑制性シナプスからの信号の強さを
引いた値がある閾値を越えると、
ニューロンは、「発火」する。

個々のシナプスの性質

同じ発火信号を受けても、受け止めるニューロン内
でのその信号の「強さ」は、シナプスごとに異なる。

ニューロン発火の条件

- あるニューロンが「発火」するかは、次のようにして決まる。

興奮性シナプスから受け取る信号の強さ全体 A から
抑制性シナプスから受け取る信号の強さ全体 B を
引いて、その値がある閾値 C より大きければ発火する。

$$A - B > C \quad \text{発火} \quad (A - B - C > 0)$$

$$A - B < C \quad \text{発火しない} \quad (A - B - C < 0)$$

- 発火賛成と発火反対の多数決。ただし、賛成票が、ある一定数以上(閾値)、反対票を上回らないといけないというルール。至極、単純である。

$$\text{賛成票} - \text{反対票} > \text{閾値} \quad \text{発火}$$

$$\text{賛成票} - \text{反対票} < \text{閾値} \quad \text{発火しない}$$

ニューロン発火の判断

賛成票 - 反対票 > 閾値 発火

X1=1

X2=1

X3=1

X4=1

X5=1

X6=1

閾値=1の場合

賛成票: $3+3+2=7$

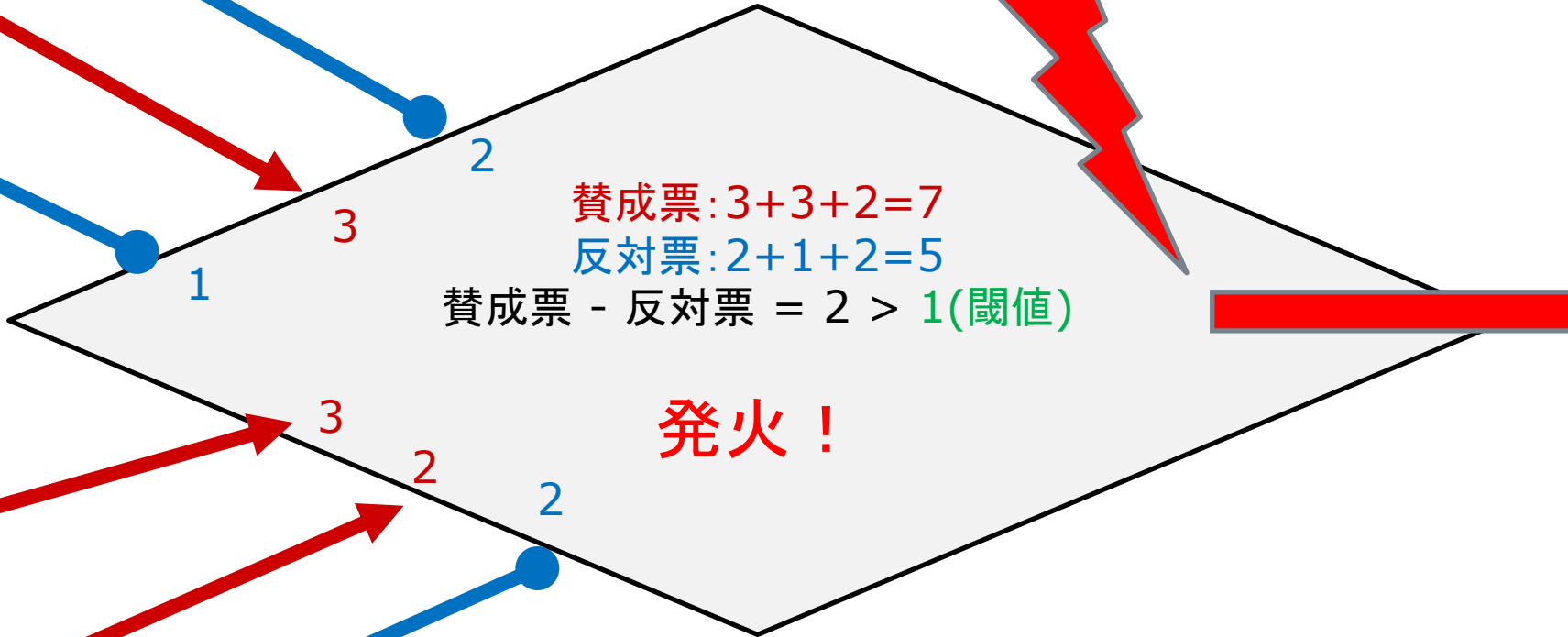
反対票: $2+1+2=5$

賛成票 - 反対票 = $2 > 1$ (閾値)

発火！

個々のシナプスの性質は異なる

内部の数字は、シナプスごとに異なる
信号を受けた時に伝えられる、信号の「強さ」

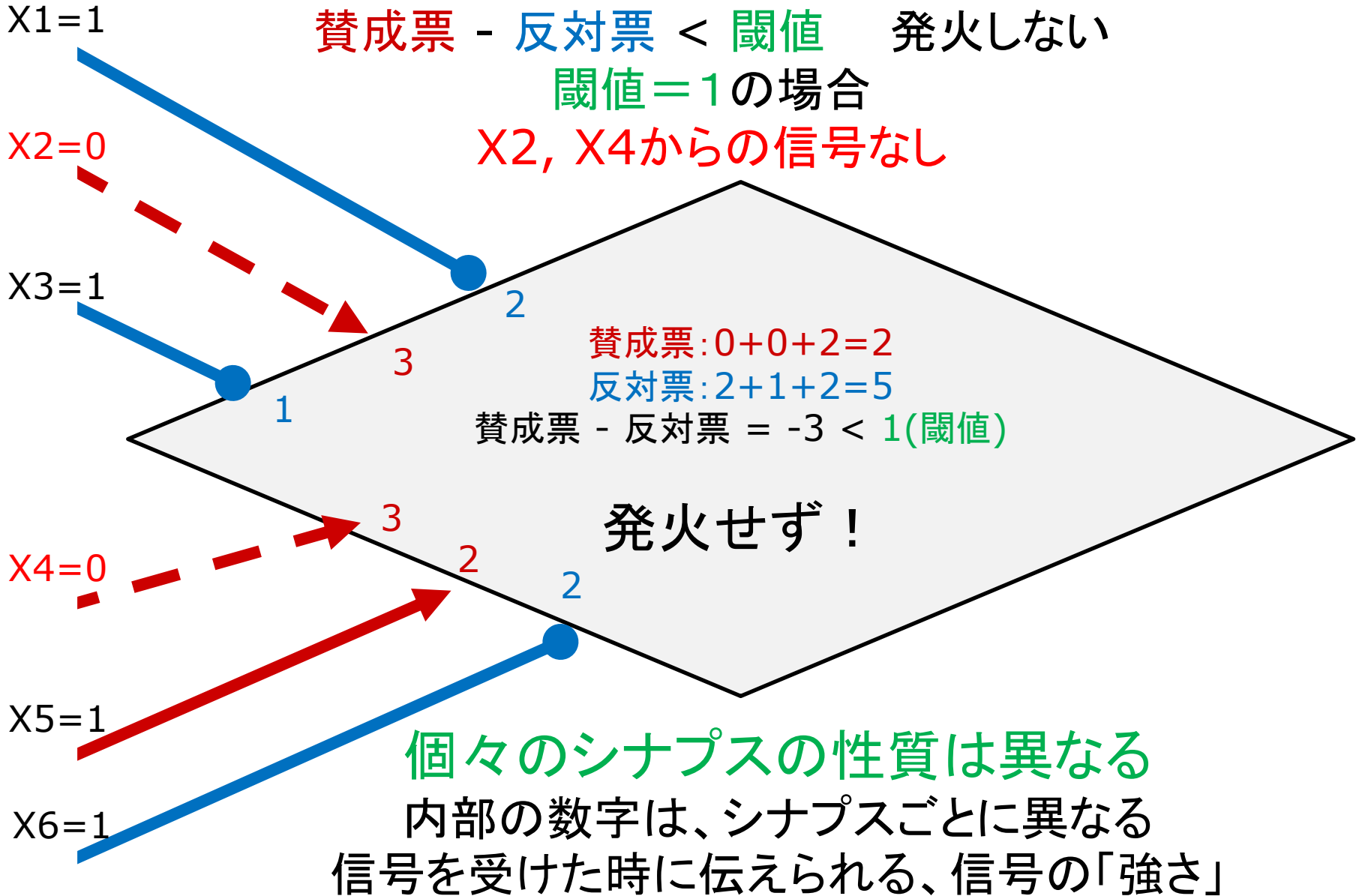


ニューロン発火の判断

賛成票 - 反対票 < 閾値 発火しない

閾値=1の場合

X2, X4からの信号なし



ニューロン発火の判断

賛成票 - 反対票 > 閾値 発火

閾値 = 1 の場合

X3, X5からの信号なし

X1=1

X2=1

X3=0

X4=1

X5=0

X6=1

1

2

3

2

2

賛成票: $3+3+0=6$

反対票: $2+0+2=4$

賛成票 - 反対票 = $2 > 1$ (閾値)

発火！

個々のシナプスの性質は異なる

内部の数字は、シナプスごとに異なる
信号を受けた時に伝えられる、信号の「強さ」

「重み」と「バイアス」

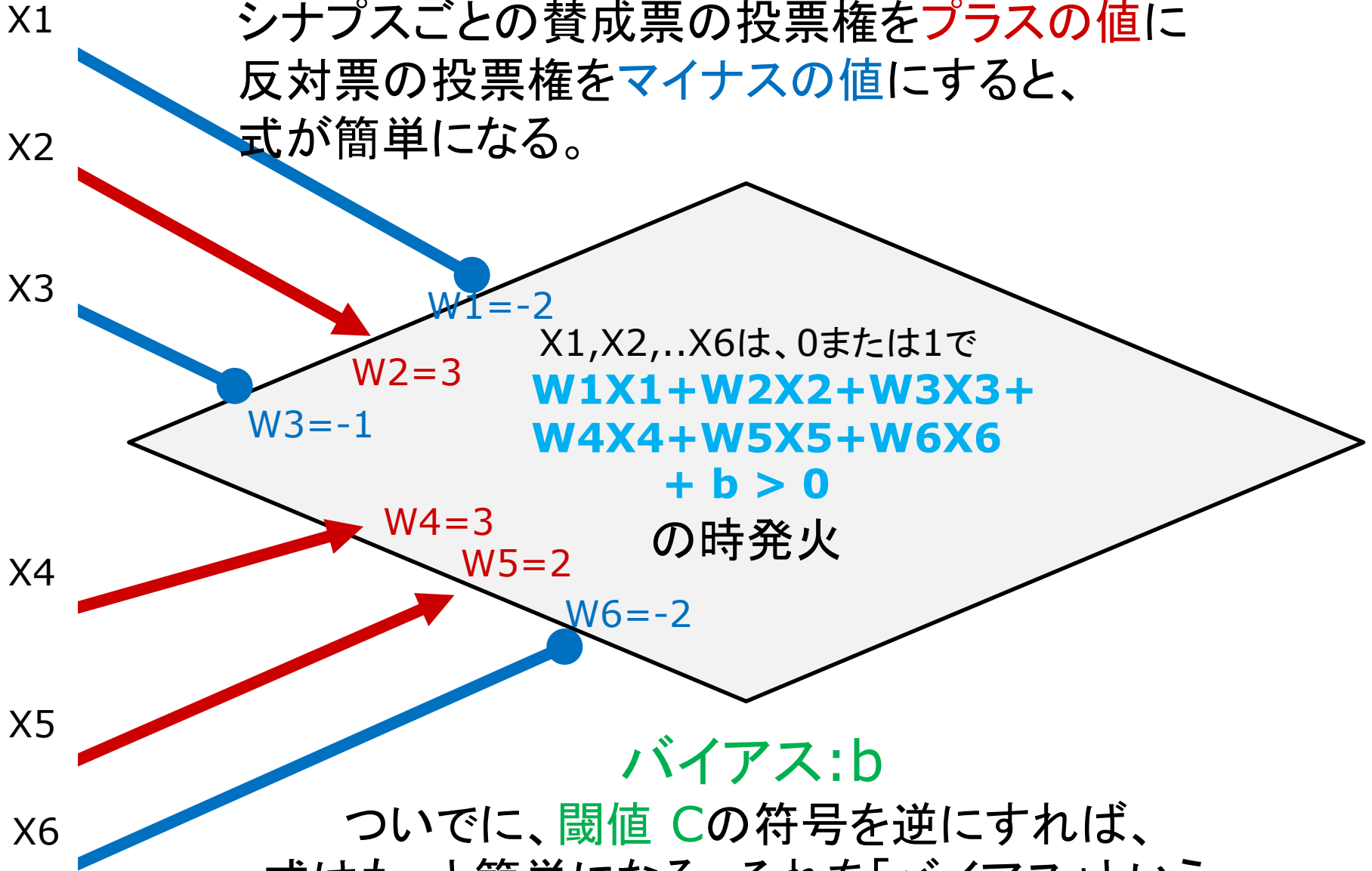
- 賛成票の数 A をプラスの値に
反対票の数 B をマイナスの値 B' にし、
閾値 C の符号を逆にして C' (バイアス) とすると、
発火の条件 $A - B > C$ (すなわち、 $A - B - C > 0$) は、
 $A + B' + C' > 0$ の形になる。
- 各シナプスごとに異なる投票権の大きさを「重み」という。
興奮性シナプスはプラスの「重み」を持ち、抑圧性シナプスはマイナスの「重み」を持つと考えるということ。
- 一つの入力 X_i (0か1の値をとる) には、それに対応した一つのシナプスが存在する。そのシナプスの重みを W_i とすれば、この一つのシナプスの投票数は、入力 X_i に 重み W_i をかけたもの。

入力 X_i と重み W_i とバイアス b で 発火の条件を表す

- 今、6つの入力 X_i ($i=1,2,\dots,6$)があったとしよう。この時、この入力に対応するシナプスの重みを W_i ($i=1,2,\dots,6$)とすれば、各シナプスでの投票数は、入力 X_i に対応する重み W_i をかけたものだから、一つのニューロン全体の投票数は、このニューロンのすべての6個のシナプスの投票を足し合わせたものになる(符号も込みで考えている)。
 - この投票数にバイアスを足し合わせたものがプラスの時、このニューロンは発火する。
 - 式で表すと、入力 $X_1, X_2, \dots, X_6$ は、0または1で
 $W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5 + W_6X_6 + b$
 > 0 の時発火ということになる。
-

重み: W_i

シナプスごとの賛成票の投票権を**プラスの値**に
反対票の投票権を**マイナスの値**にすると、
式が簡単になる。



バイアス: b

ついでに、**閾値 C** の符号を逆にすれば、
式はもっと簡単になる。それを「バイアス」という。

重み $W_i = [-2, 3, -1, 3, 2, -2]$

バイアス $b = -1$

入力 $X_i = [1, 0, 1, 0, 1, 1]$
の場合

$X_1 = 1$

$X_2 = 0$

$X_3 = 1$

$X_4 = 0$

$X_5 = 1$

$X_6 = 1$

$$W_1X_1 + W_2X_2 + W_3X_3 + \\ -2 W_4X_4 + W_5X_5 + W_6X_6 \\ + b =$$

$$(-2)x_1 + 3x_0 + (-1)x_1 + \\ 3x_0 + 2x_1 + (-2)x_1 \\ -1 =$$

$$-2 - 2 + 2 - 2 - 1 = -5 < 0$$

-2 発火せず！

-1

3

3

2

重み $W_i = [-2, 3, -1, 3, 2, -2]$

バイアス $b = -1$

入力 $X_i = [1, 1, 0, 1, 0, 1]$
の場合

$X_1 = 1$

$X_2 = 1$

$X_3 = 0$

$X_4 = 1$

$X_5 = 0$

$X_6 = 1$

$$W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5 + W_6X_6 + b =$$

$$(-2) \times 1 + 3 \times 1 + (-1) \times 0 + 3 \times 1 + 2 \times 0 + (-2) \times 1$$

$-1 =$

$$-2 + 3 + 3 - 2 = 2 > 0$$

発火！

3

-1

3

2

-2

一つのニューロンの 内部の状態の表現

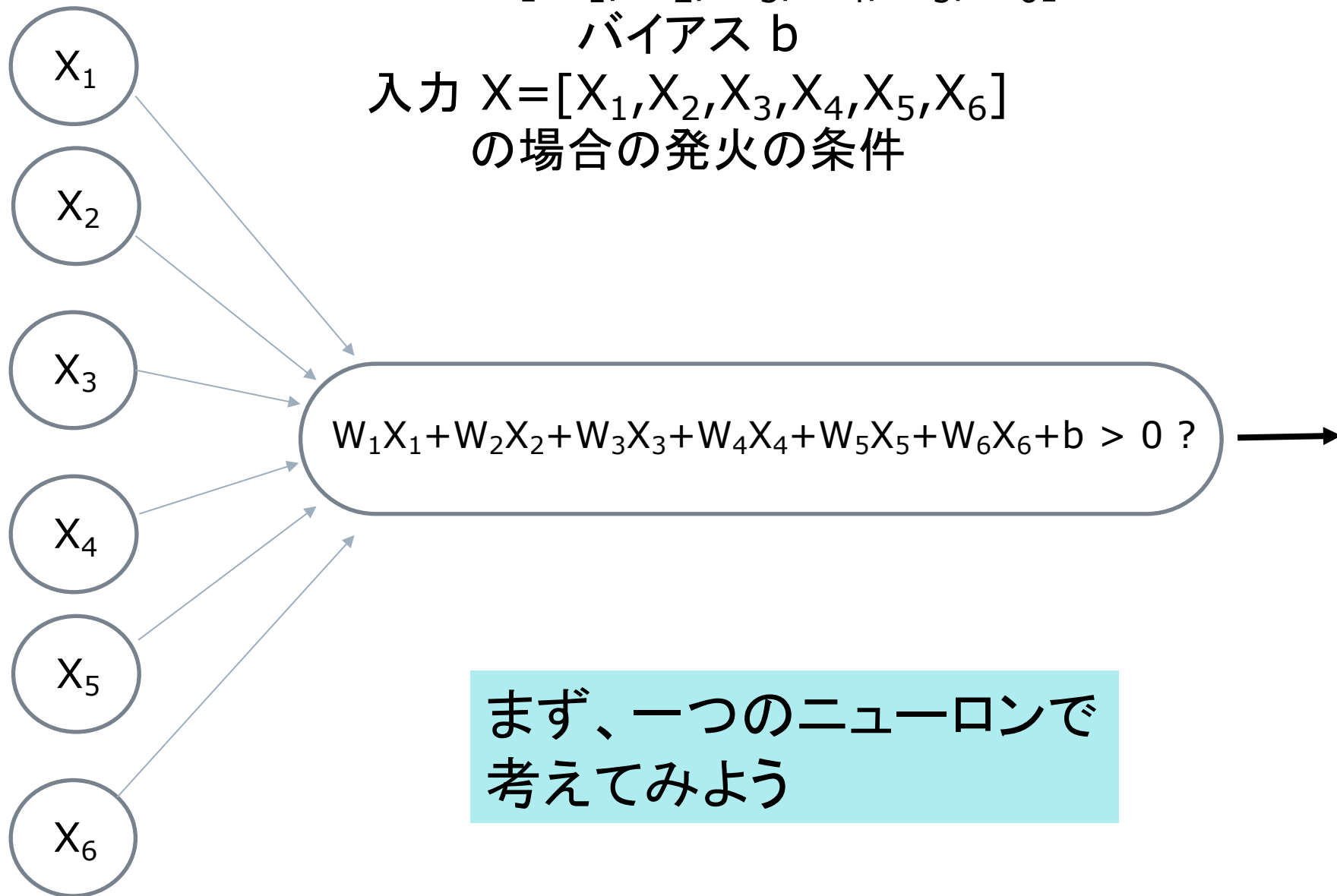
一つのニューロンの内部の状態は、ニューロンがシナプスを通じて受け取る刺激である「入力」と、シナプスごとの「重み」と、発火の閾値に対応する「バイアス」の三つの量で表現できる。

重み $W=[W_1, W_2, W_3, W_4, W_5, W_6]$

バイアス b

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$

の場合の発火の条件



まず、一つのニューロンで
考えてみよう

X_1

X_2

X_3

X_4

X_5

X_6

重み $W = [W_1, W_2, W_3, W_4, W_5, W_6]$

入力 $X = [X_1, X_2, X_3, X_4, X_5, X_6]$

バイアス b

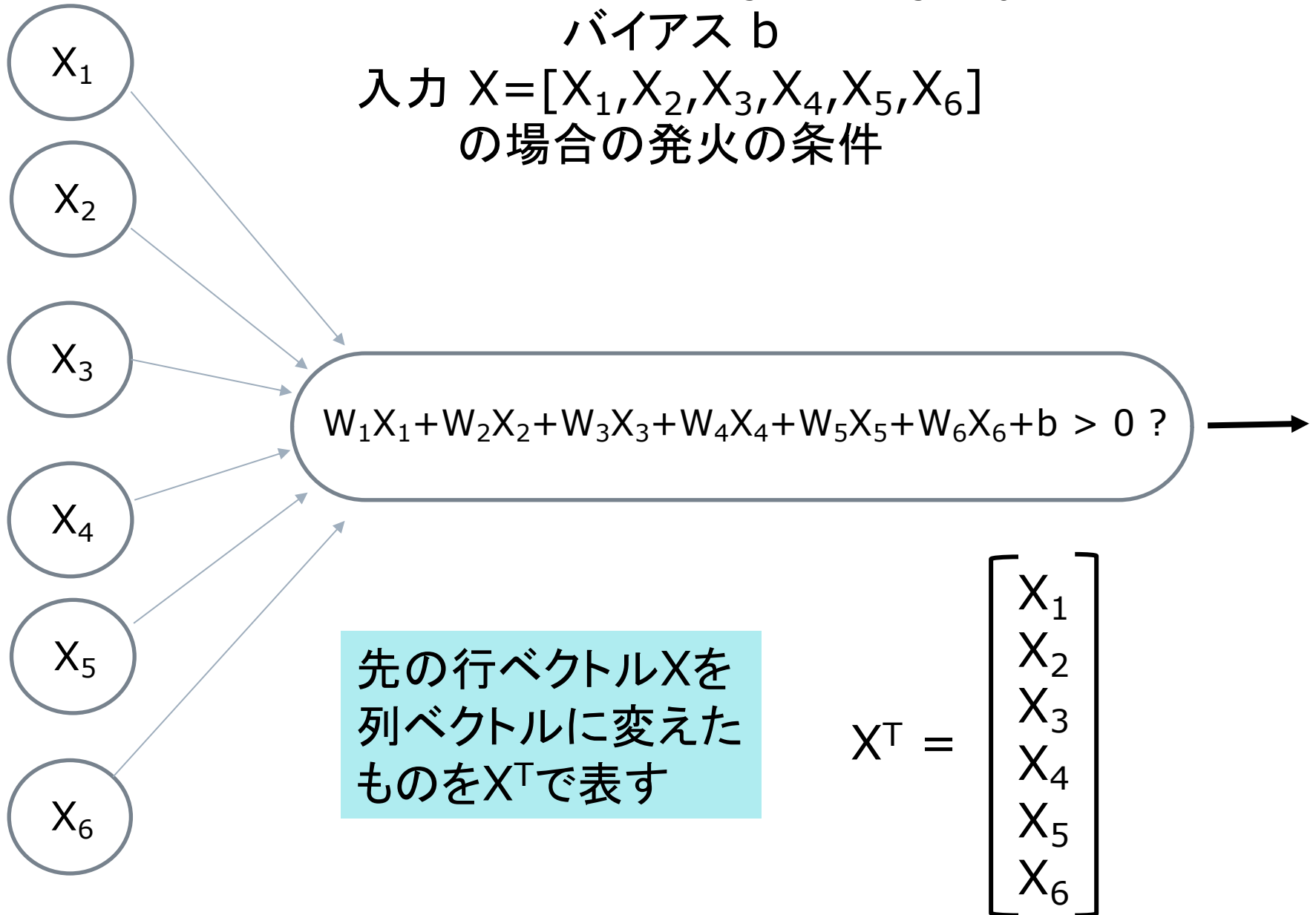
$$W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5 + W_6X_6 + b > 0 ?$$

重みと入力の対応する項目同士を
かけて、すべて足し合わせて、それに
バイアス b を足したもの。

重み $W=[W_1, W_2, W_3, W_4, W_5, W_6]$

バイアス b

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$
の場合の発火の条件



行ベクトルと列ベクトルの積

□ 行ベクトル $W=[W_1, W_2, W_3, W_4, W_5, W_6]$ と
行ベクトル $X=[X_1, X_2, X_3, X_4, X_5, X_6]$ を列ベクトルに変えた X^T との積を次のように定義する。

□ $W \cdot X^T =$

$$[W_1, W_2, W_3, W_4, W_5, W_6] \cdot \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{bmatrix}$$

$$= W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5 + W_6X_6$$

□ 対応する要素を掛けて、足し合わせたものである。

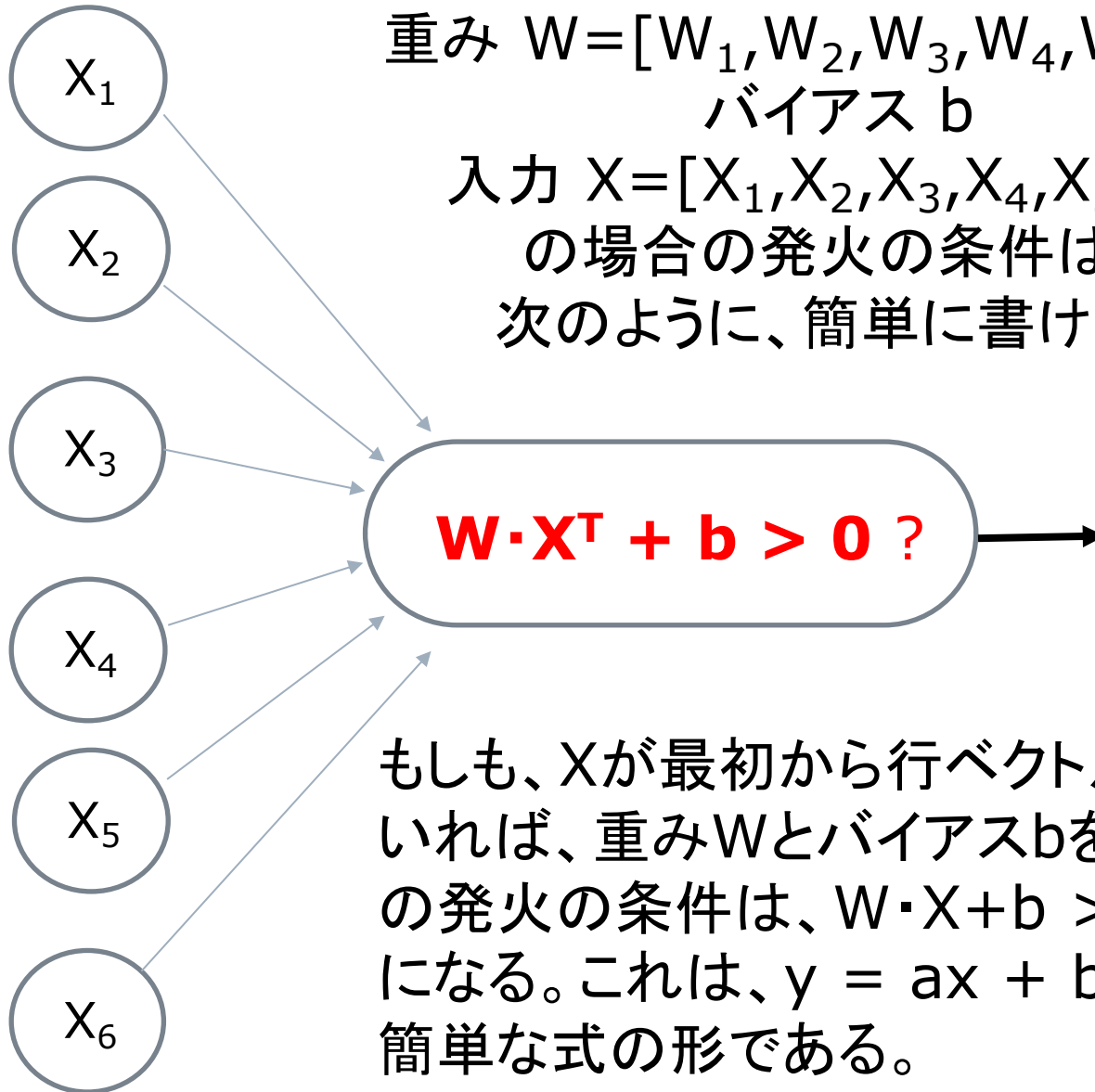
この時、

重み $W=[W_1, W_2, W_3, W_4, W_5, W_6]$

バイアス b

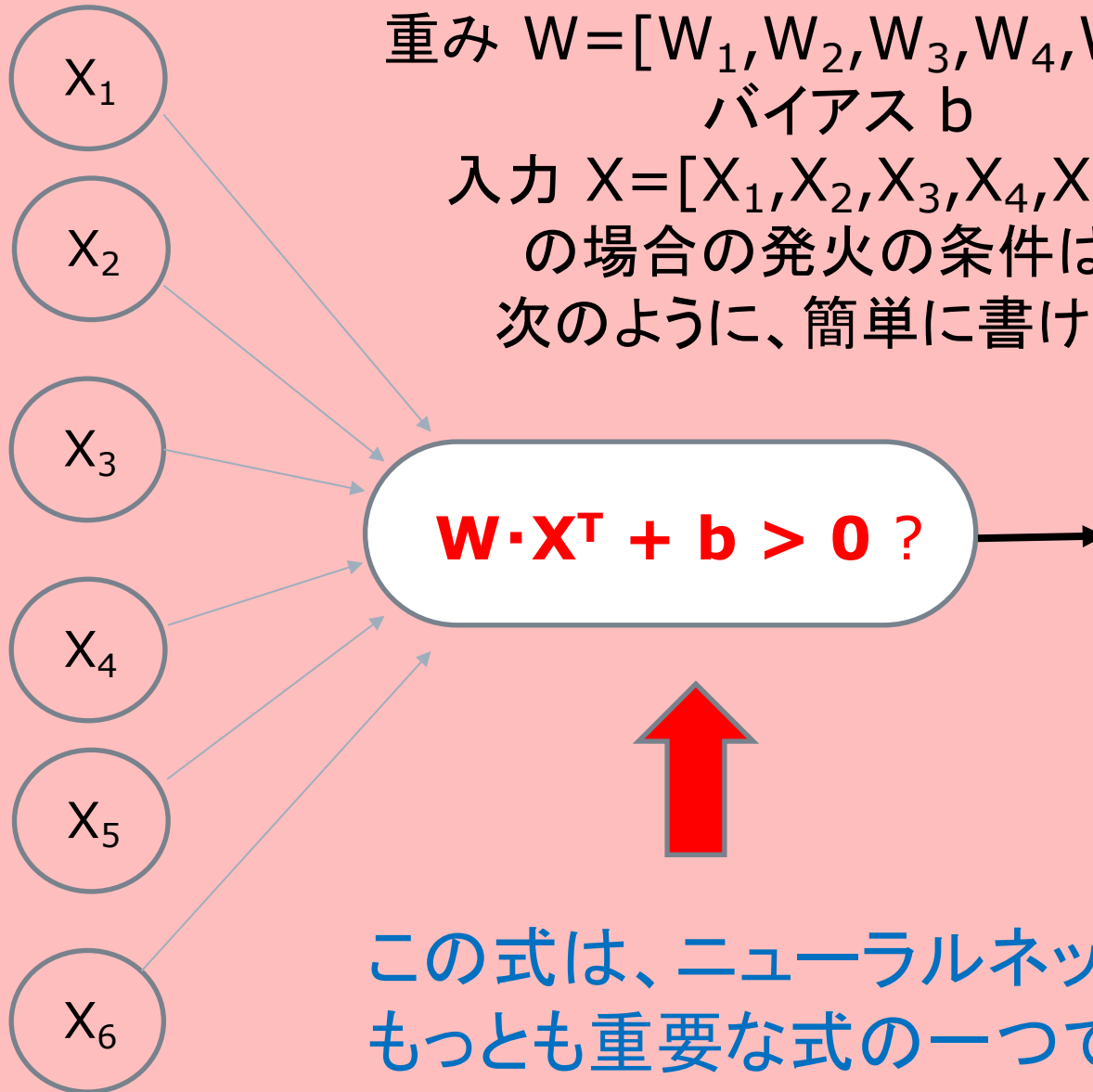
入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$

の場合の発火の条件は、
次のように、簡単に書ける。



もしも、 X が最初から行ベクトルの形で与えられて
いれば、重み W とバイアス b を持つあるニューロン
の発火の条件は、 $W \cdot X + b > 0$ で与えられること
になる。これは、 $y = ax + b > 0$ と同じくらい、
簡単な式の形である。

この時、
重み $W=[W_1, W_2, W_3, W_4, W_5, W_6]$
バイアス b
入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$
の場合の発火の条件は、
次のように、簡単に書ける。



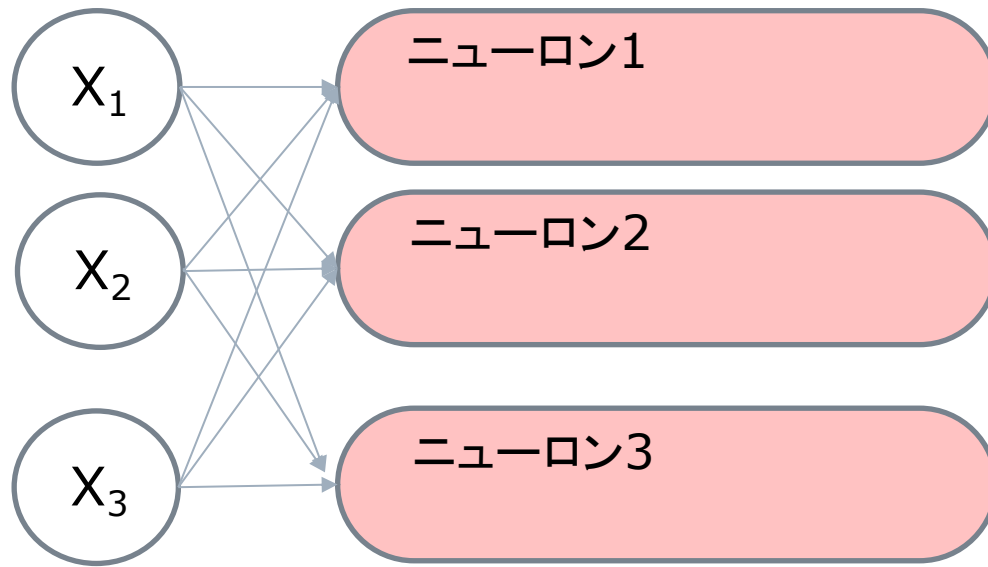
この式は、ニューラルネットワークで、
もっとも重要な式の一つである。

複数のニューロンからなる一つの層の内部の状態の表現

複数のニューロンの集まりが、一つの「層」として、入力を受け取り出力を返すことがある。

この層の内部の状態も、この層が受け取る「入力の全体」と、シナプスごとの「重みの全体」と、層を構成するニューロンの閾値に対応する「バイアスの全体」という三つの量で表現できる。

入力 $X=[X_1, X_2, X_3]$



三つのニューロンで
考えてみよう

例えば、入力 $X=[1,0,1]$ で

ニューロン1の重みが $[2, -3, 4]$ バイアスが -4 なら

ニューロン1の出力は、 $2 \cdot 1 + (-3) \cdot 0 + 4 \cdot 1 - 4 = 2 > 0$ で発火。

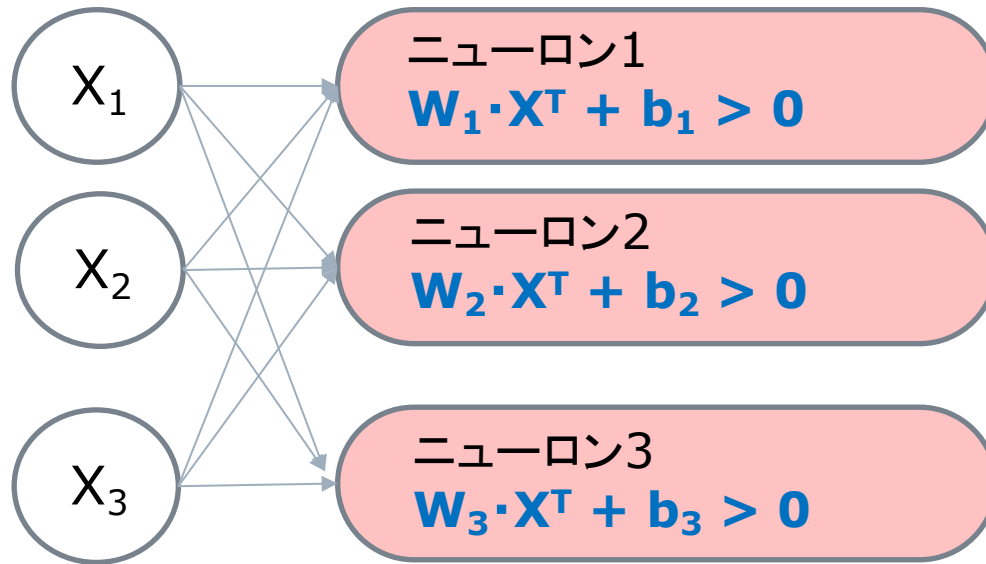
ニューロン2の重みが $[-4, 1, -5]$ バイアスが 5 なら

ニューロン2の出力は、 $(-4) \cdot 1 + 1 \cdot 0 + (-5) \cdot 1 + 5 = -4 < 0$ で発火せず。

ニューロン3の重みが $[-4, -2, 5]$ バイアスが 2 なら

ニューロン3の出力は、 $(-4) \cdot 1 + (-2) \cdot 0 + 5 \cdot 1 + 2 = 3 > 0$ で発火。

入力 $X=[X_1, X_2, X_3]$



重み $\mathbf{W}_1=[W_{11}, W_{12}, W_{13}]$
バイアス b_1

重み $\mathbf{W}_2=[W_{21}, W_{22}, W_{23}]$
バイアス b_2

重み $\mathbf{W}_3=[W_{31}, W_{32}, W_{33}]$
バイアス b_3

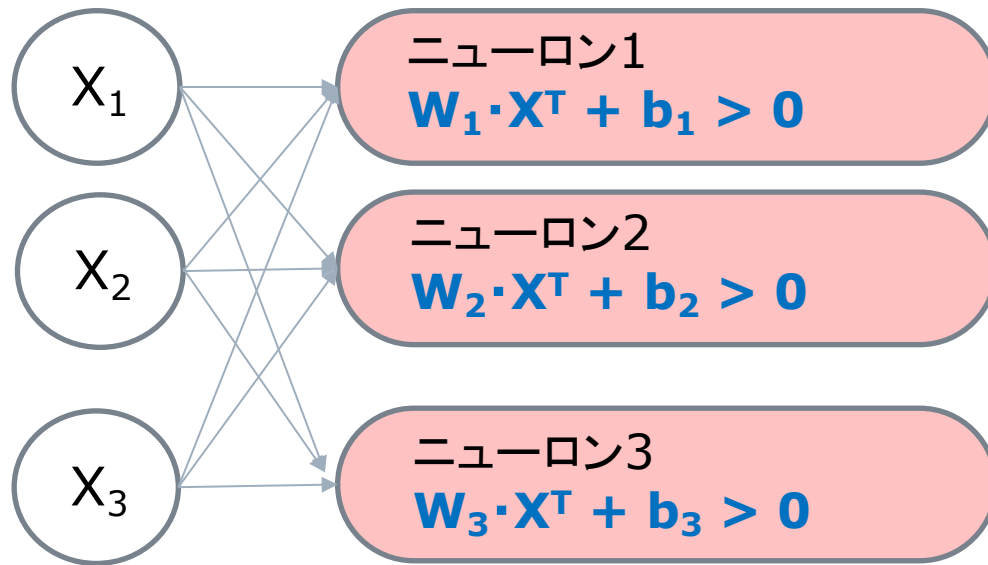
一般的には、

ニューロン1の値: $\mathbf{W}_1 \cdot \mathbf{X}^T + b_1 = W_{11}X_1 + W_{12}X_2 + W_{13}X_3 + b_1$

ニューロン2の値: $\mathbf{W}_2 \cdot \mathbf{X}^T + b_2 = W_{21}X_1 + W_{22}X_2 + W_{23}X_3 + b_2$

ニューロン3の値: $\mathbf{W}_3 \cdot \mathbf{X}^T + b_3 = W_{31}X_1 + W_{32}X_2 + W_{33}X_3 + b_3$

入力 $X=[X_1, X_2, X_3]$



重み $W_1=[W_{11}, W_{12}, W_{13}]$
バイアス b_1

重み $W_2=[W_{21}, W_{22}, W_{23}]$
バイアス b_2

重み $W_3=[W_{31}, W_{32}, W_{33}]$
バイアス b_3

これを、行列の積と和を使うと（あとで少し述べる）

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} W_{11}X_1 + W_{12}X_2 + W_{13}X_3 + b_1 \\ W_{21}X_1 + W_{22}X_2 + W_{23}X_3 + b_2 \\ W_{31}X_1 + W_{32}X_2 + W_{33}X_3 + b_3 \end{bmatrix}$$

これも、 $W \cdot X^T + b$ の形をしている！

ニューロンが発火する条件

$$W \cdot X^T + b > 0$$

□ 一つのニューロンで、一つの入力の場合。

一つの入力を X (実数)、そのシナプスの重みを W (実数)、ニューロンのバイアスを b (実数)とすると、条件は

$$W \cdot X + b > 0$$

$W \cdot X$ は、普通の掛け算

□ 一つのニューロンで、複数の入力の場合。

複数の入力を $\mathbf{X}$ (ベクトル)、その複数のシナプスの重みを $\mathbf{W}$ (ベクトル)、ニューロンのバイアスを b (実数)とすると、条件は

$$\mathbf{W} \cdot \mathbf{X}^T + b > 0$$

$\mathbf{W} \cdot \mathbf{X}^T$ は、ベクトルの内積

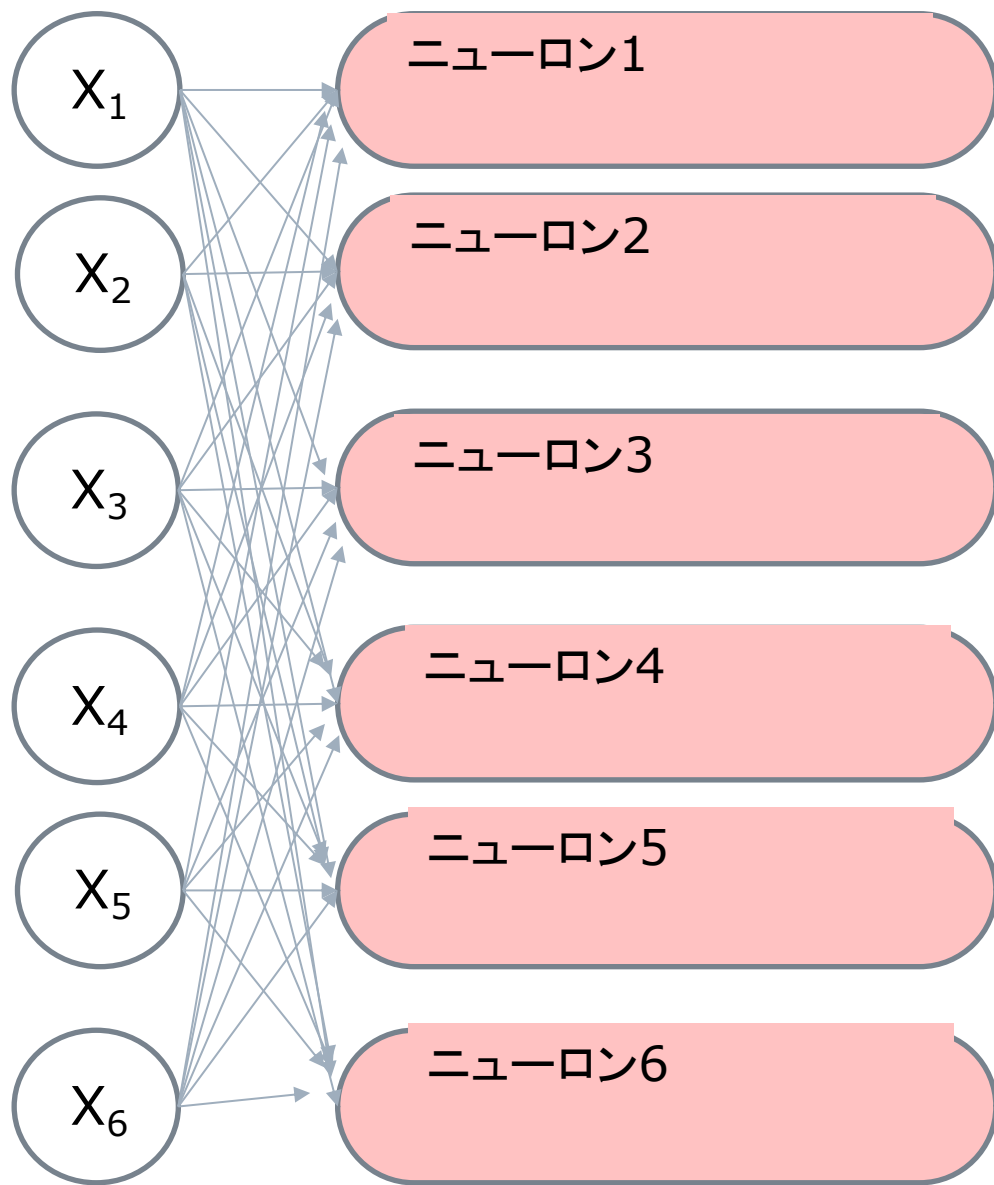
□ 複数のニューロンの層で、複数の入力の場合

複数の入力を $\mathbf{X}$ (ベクトル)、その層内のシナプスの重み全体を $\mathbf{W}$ (行列)、ニューロンの複数のバイアスを $\mathbf{b}$ (ベクトル)とすると、条件は

$$\mathbf{W} \cdot \mathbf{X}^T + \mathbf{b} > 0$$

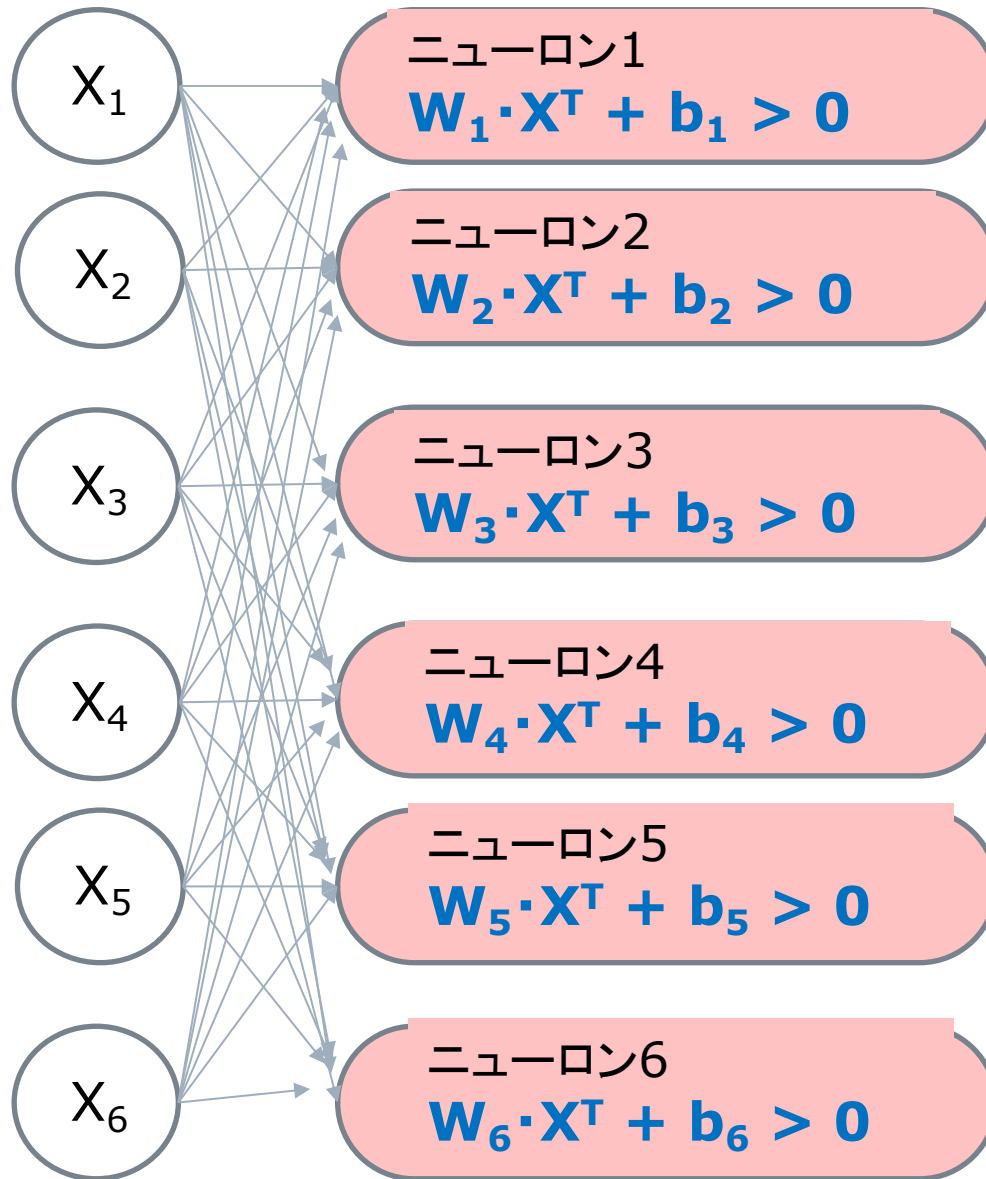
$\mathbf{W} \cdot \mathbf{X}^T$ は、行列とベクトルの積

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$



六つのニューロンで
考えてみよう。
基本的には、同じで
ある。

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$



重み $W_1=[W_{11}, W_{12}, W_{13}, W_{14}, W_{15}, W_{16}]$
バイアス b_1

重み $W_2=[W_{21}, W_{22}, W_{23}, W_{24}, W_{25}, W_{26}]$
バイアス b_2

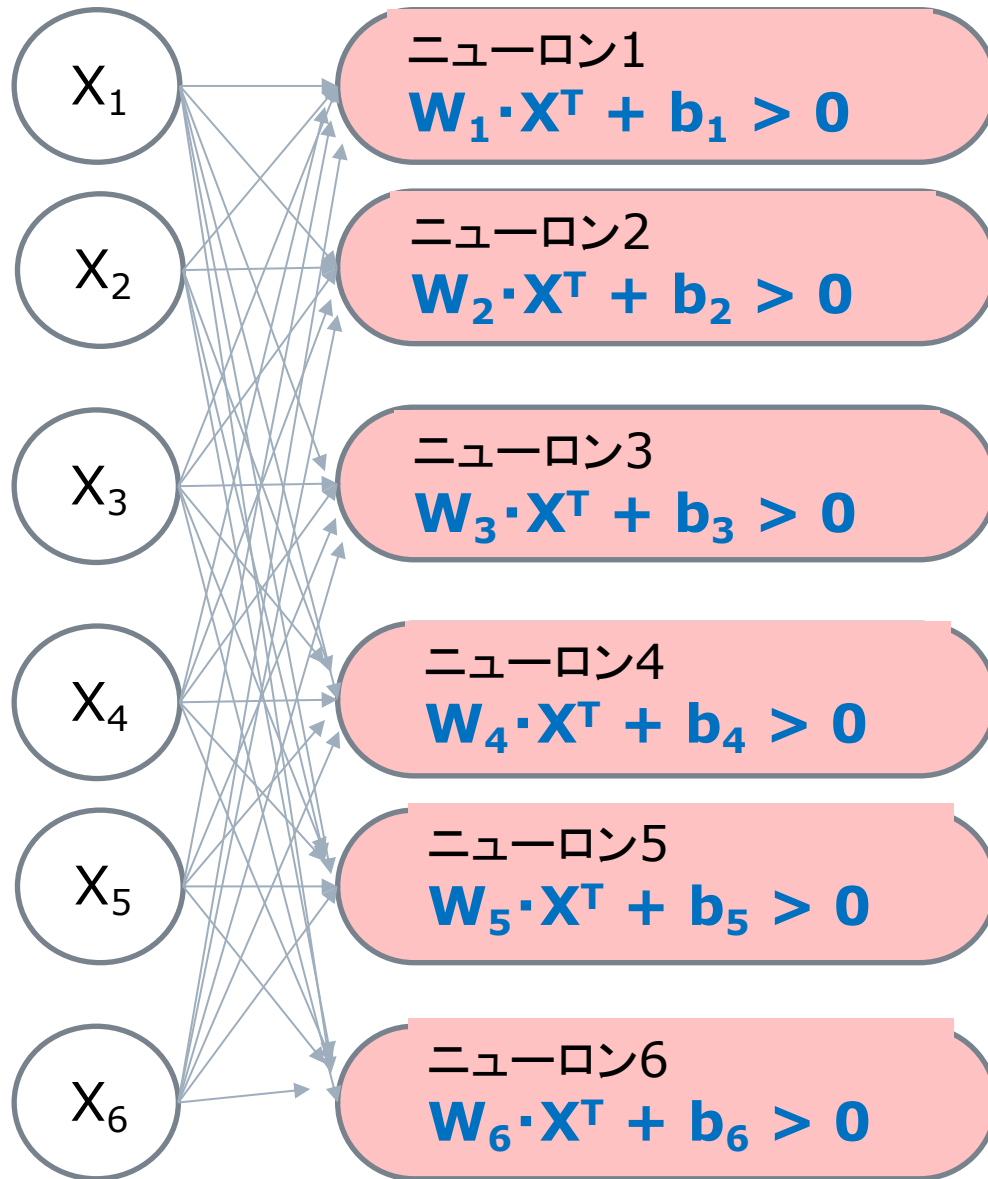
重み $W_3=[W_{31}, W_{32}, W_{33}, W_{34}, W_{35}, W_{36}]$
バイアス b_3

重み $W_4=[W_{41}, W_{42}, W_{43}, W_{44}, W_{45}, W_{46}]$
バイアス b_4

重み $W_5=[W_{51}, W_{52}, W_{53}, W_{54}, W_{55}, W_{56}]$
バイアス b_5

重み $W_6=[W_{61}, W_{62}, W_{63}, W_{64}, W_{65}, W_{66}]$
バイアス b_6

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$



値: ~~$W_{11}X_1 + W_{12}X_2 + W_{13}X_3 +$~~
 $W_{14}X_4 + W_{15}X_5 + W_{16}X_6 + b_1$

値: $W_{21}X_1 + W_{22}X_2 + W_{23}X_3 +$
 $W_{24}X_4 + W_{25}X_5 + W_{26}X_6 + b_2$

値: $W_{31}X_1 + W_{32}X_2 + W_{33}X_3 +$
 $W_{34}X_4 + W_{35}X_5 + W_{36}X_6 + b_3$

値: $W_{41}X_1 + W_{42}X_2 + W_{43}X_3 +$
 $W_{44}X_4 + W_{45}X_5 + W_{46}X_6 + b_4$

値: ~~$W_{51}X_1 + W_{52}X_2 + W_{53}X_3 +$~~
 $W_{54}X_4 + W_{55}X_5 + W_{56}X_6 + b_5$

値: ~~$W_{61}X_1 + W_{62}X_2 + W_{63}X_3 +$~~
 $W_{64}X_4 + W_{65}X_5 + W_{66}X_6 + b_6$

行列による表現(六個のニューロンの場合)

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} & W_{15} & W_{16} \\ W_{21} & W_{22} & W_{23} & W_{24} & W_{25} & W_{26} \\ W_{31} & W_{32} & W_{33} & W_{34} & W_{35} & W_{36} \\ W_{41} & W_{42} & W_{43} & W_{44} & W_{45} & W_{46} \\ W_{51} & W_{52} & W_{53} & W_{54} & W_{55} & W_{56} \\ W_{61} & W_{62} & W_{63} & W_{64} & W_{65} & W_{66} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} =$$

$$\begin{bmatrix} W_{11}X_1 + W_{12}X_2 + W_{13}X_3 + W_{14}X_4 + W_{15}X_5 + W_{16}X_6 + b_1 \\ W_{21}X_1 + W_{22}X_2 + W_{23}X_3 + W_{24}X_4 + W_{25}X_5 + W_{26}X_6 + b_2 \\ W_{31}X_1 + W_{32}X_2 + W_{33}X_3 + W_{34}X_4 + W_{35}X_5 + W_{36}X_6 + b_3 \\ W_{41}X_1 + W_{42}X_2 + W_{43}X_3 + W_{44}X_4 + W_{45}X_5 + W_{46}X_6 + b_4 \\ W_{51}X_1 + W_{52}X_2 + W_{53}X_3 + W_{54}X_4 + W_{55}X_5 + W_{56}X_6 + b_5 \\ W_{61}X_1 + W_{62}X_2 + W_{63}X_3 + W_{64}X_4 + W_{65}X_5 + W_{66}X_6 + b_6 \end{bmatrix}$$

これも、 **$W \cdot X^T + b$** の形をしている！

式は便利だ！

$Y = W \cdot X^T + b$ が表すもの

数学的な表記を使うと、問題を簡潔に定式化できる。
ここでは、一つのニューロンの状態だけでなく、複数のニューロンからなる層も、一つの式で表現できる背景について述べる。

式は便利だ！

$Y = W \cdot X^T + b$ が表すもの

- ニューラル・ネットワークの本を読んで、最初につまずくのは、行列が出てくるあたりだと思う。ただ、ニューラル・ネットワークの理解で、当面、必要なことは、式 $Y = W \cdot X^T + b$ が表現しているものを理解することだ。見慣れていないだけで、難しい数学ではない。
- 具体的な例で、式 $Y = W \cdot X^T + b$ が表しているものを、紹介したいと思う。一見、複雑に見える沢山の関係が、この式一つで表現されていることがわかるはずだ。式は便利だということがわかってもらえると思う。
- 以下の例では、 X も Y も b も、最初から列ベクトルだとしよう（このことを、仮に、 $X = [X_1, X_2, \dots, X_n]^T$ のように表そう）。この時、 $Y = W \cdot X + b$ が、何を表すかを考えよう。

$Y = W \cdot X + b$ が表すもの W が数値(スカラー)の時

- $Y=y, X=x, W=a, b=b$ なら、この式は、
 $y = ax + b$ という、式を表す。
- $Y=y, X=x, W=2, b=3$ なら、この式は、
 $y = 2x + 3$ という、式を表す。
- $Y=y, X=x, W=4, b=0$ なら、この式は、
 $y = 4x$ という、式を表す。

$Y = W \cdot X + b$ が表すもの W がベクトルの時

- $Y=y, X=[x_1, x_2]^T, W=[w_1, w_2], b=b_0$ なら、この式は、
 $y = w_1 \cdot x_1 + w_2 \cdot x_2 + b_0$ という、式を表す。
- $Y=y, X=[c, d]^T, W=[m, n], b=b$ なら、この式は、
 $y = m \cdot c + n \cdot d + b$ という、式を表す。
- $Y=y, X=[1, 0]^T, W=[2, 3], b=4$ なら、この式は、
 $y = 2 \cdot 1 + 3 \cdot 0 + 4 = 6$ という、式を表す。
- $Y=y, X=[x_1, x_2, x_3]^T, W=[w_1, w_2, w_3], b=b_0$ なら、
 $y = w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + b_0$
- $Y=y, X=[x_1, x_2, x_3, x_4]^T, W=[w_1, w_2, w_3, w_4], b=b_0$ なら、
 $y = w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + w_4 \cdot x_4 + b_0$

$Y = W \cdot X + b$ が表すもの Wが行列の時

- 2x2の行列(2行2列の行列)を、 $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ と表そう。
同様に、3x3 (3行3列の行列)の行列を、 $\begin{bmatrix} d & e & f \\ g & h & i \\ j & k & l \end{bmatrix}$ と表そう。
- $Y = [y_1, y_2]^T$, $X = [x_1, x_2]^T$, $W = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$,
 $b = [1, 2]^T$ のとき、 $Y = W \cdot X + b$ は、次の二つの式を表す。
 $y_1 = 2x_1 + 3x_2 + 1$, $y_2 = 4x_1 + 5x_2 + 2$
- $Y = [y_1, y_2, y_3]^T$, $X = [x_1, x_2, x_3]^T$, $W = \begin{bmatrix} 2 & 3 & -1 \\ 4 & -5 & 1 \\ 1 & 2 & 3 \end{bmatrix}$, $b = [1, 2, 3]^T$ のとき、 $Y = W \cdot X + b$ は、次の三つの式を表す。
 $y_1 = 2x_1 + 3x_2 - x_3 + 1$, $y_2 = 4x_1 - 5x_2 + x_3 + 2$, $y_3 = x_1 + 2x_2 + 3x_3 + 3$

$Y = W \cdot X + b$ が表すもの Wが行列の時

- 例えば、Wが6x6(6行6列)の行列の時、 $Y = W \cdot X + b$ が表すものの例については、先に見た。もちろんnが100でも10,000でも同じ式で、表現できる。
- Wが n行 m列の行列の時、 $Y = W \cdot X + b$ が、n個の式を表していることを示すために、次のように書くこともある。
 $Y_n = W_n \cdot X + b_n$
- こうした表記は、数百・数千のノードからなるニューラル・ネットワークが満たす多数の関係式を、一つの式で簡潔にまとめることができるので、プログラミング上も極めて有用である。

$Y=W \cdot X^T+b$ と $Y=X \cdot W+b$ という 二つの表記

- これまで、入力の $X=[X_1, X_2, \dots, X_n]$ を行ベクトルとして、 $Y=W \cdot X^T+b$ という表記を用いてきた。多くのドキュメントもこうした表記を採用している。
- $X=[X_1, X_2, \dots, X_n]^T$ として、 $Y=W \cdot X+b$ と表しても、この二つの表記は、同じ内容を表し、重みを表す行列 W も、同じものである。どちらでも X^T , b は列ベクトルである。
- ただ、 $X=[X_1, X_2, \dots, X_n]$ を行ベクトルとした時、 $Y=X \cdot W+b$ という表記も可能である。ここでの W は、先の W の行と列を入れ替えたものである。 $W=W^T$ 。 b は、行ベクトルとなる。 $b=b^T$ である。基本的には同じである。
- ただ、TensorFlowでは、 $Y=X \cdot W+b$ の方の表記を採用している。TensorFlowに関連する部分に入ったら、この表記に切り替えたいと思う。

ニューロンを動かす

-- 活性化関数 (Activator)

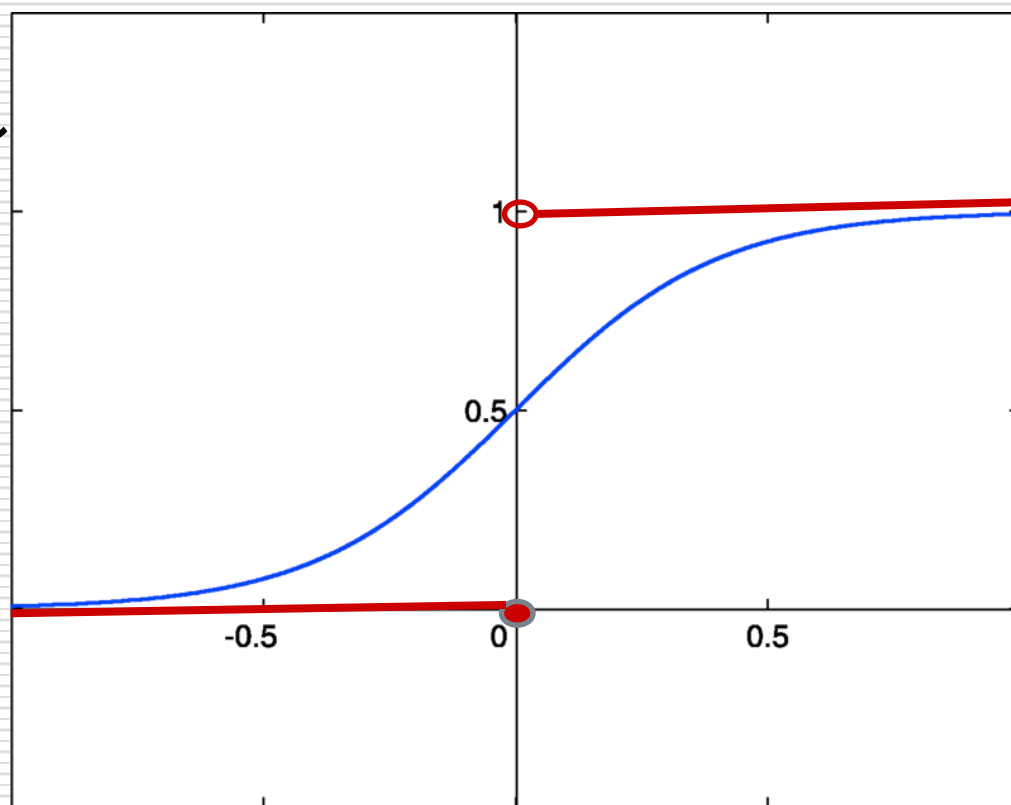
活性化関数は、その名前の通りに、一つのニューロンが発火(活性化)する時の条件と出力を関数の形で表したものである。いろいろな種類の活性化関数がある。

活性化関数

- 先に見たようにように、重み W とバイアス b を用いて、 $W \cdot X + b$ という式で、ニューロンの興奮のレベルが計算できるとするなら、「 $W \cdot X + b > 0$ なら、ニューロンは発火する」という条件は、どのように表現できるだろうか？
- $W \cdot X + b$ は、ニューロンの内部の状態を表す量なので、その値を、ニューロンの出力に変える関数 ϕ をうまく定義して、 $\phi(W \cdot X + b)$ がそのニューロンの発火・出力を表現するようにすればいい。こうした関数 ϕ を、「活性化関数」とよぶ。
- 例えば、ニューロンの発火の信号の強さが 1 で、発火しない時は信号が 0 であるなら、次のような関数 ϕ を考えればいい。
$$\phi(x) = \begin{cases} 0 & x \leq 0 \text{ の時} \\ 1 & x > 0 \text{ の時} \end{cases}$$

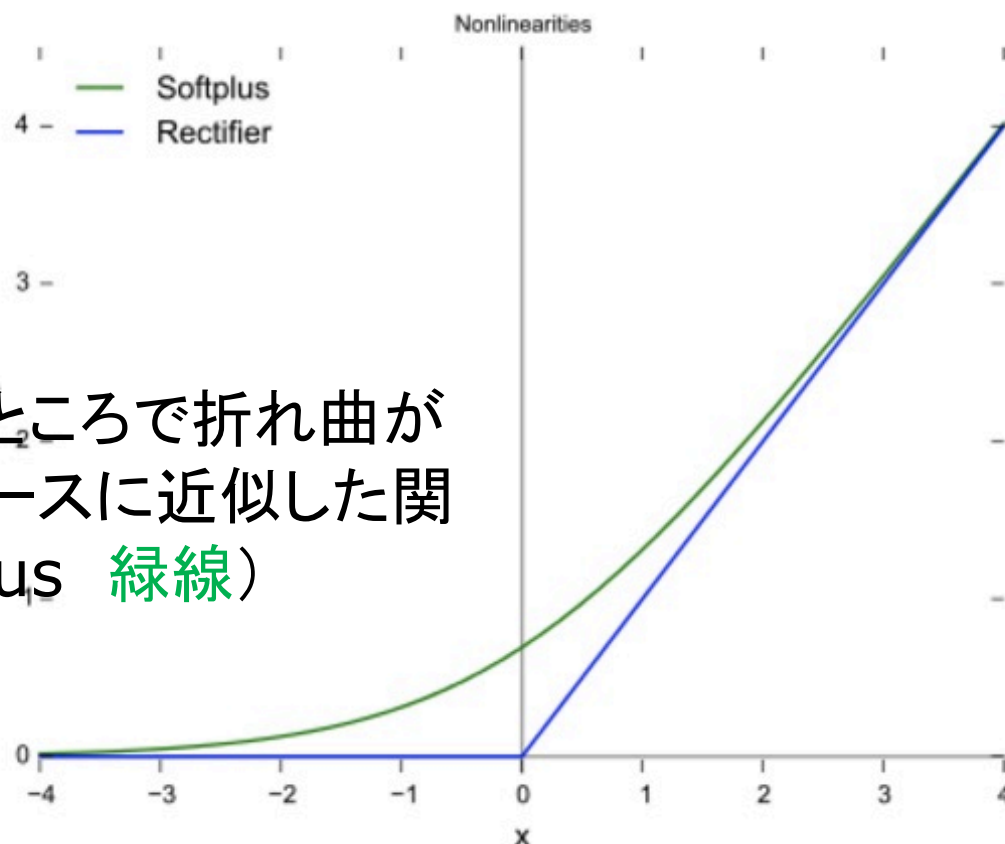
sigmoid 関数 (logistic 関数)

- 先の φ を、右のグラフでは、赤線で示した。 $x=0$ の時に、非連続に変化する。
- 実際の応用では、こうした飛躍のない、青線のような関数が持ちいられることが多い。
- これをsigmoid関数という。(logistic関数)
- 近似的には、
 $x < 0$ の時 0
 $x > 0$ の時 1
基本的には0と1の間の値をとる。



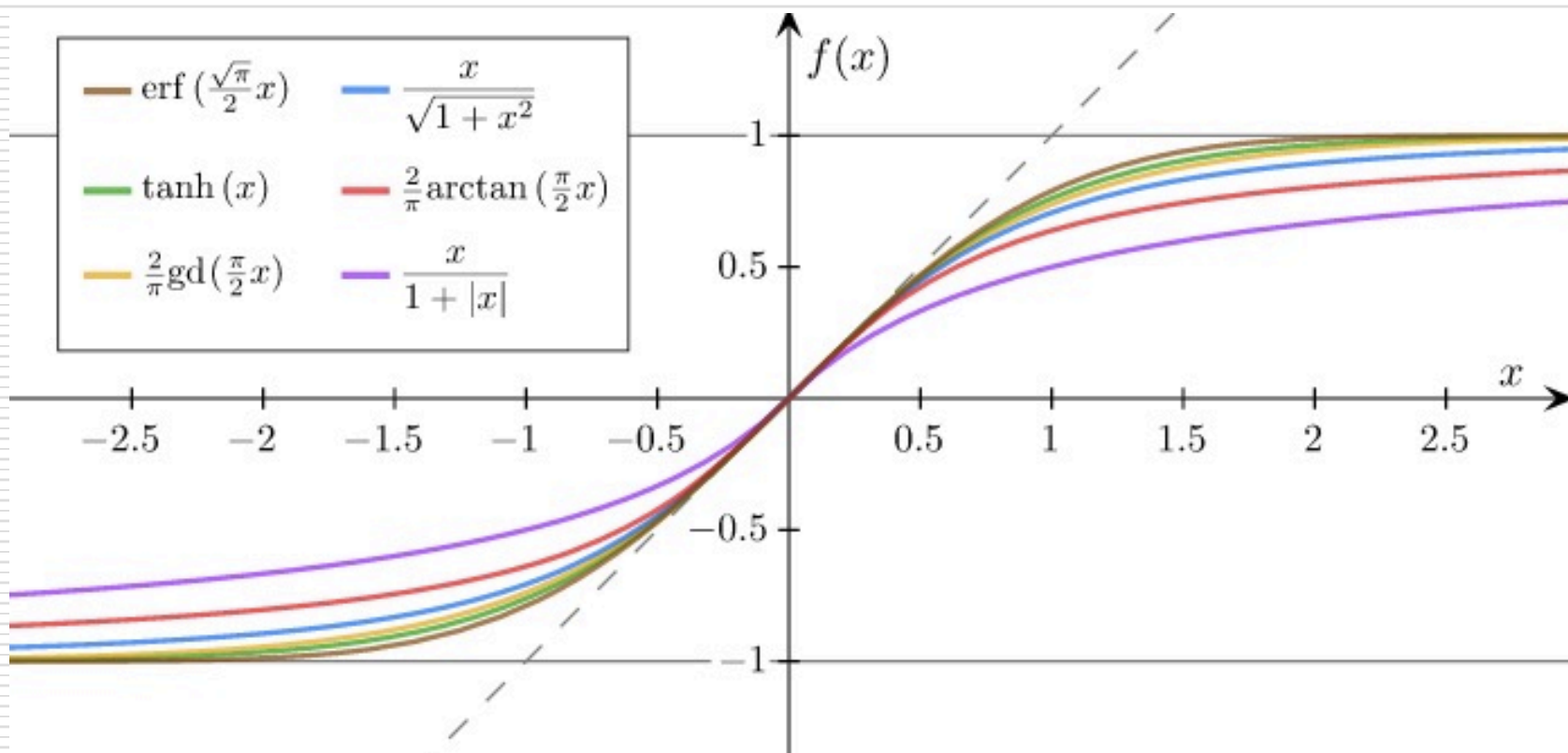
ReLU (rectified linear unit)

- rectifier は、「整流器」。ReLUは、マイナスの値は通さず0を返すが、プラスの値は、そのまま通す。(青線)
- sigmoidは、0と1の間の値を出力するが、ReLUの出力には、そうした制限はない。
- 断続はないが、 $x=0$ のところで折れ曲がっている。それを、スムーズに近似した関数も存在する。(Softplus 緑線)
- $x < 0$ の時 0
 $x > 0$ の時 x
の値をとる。



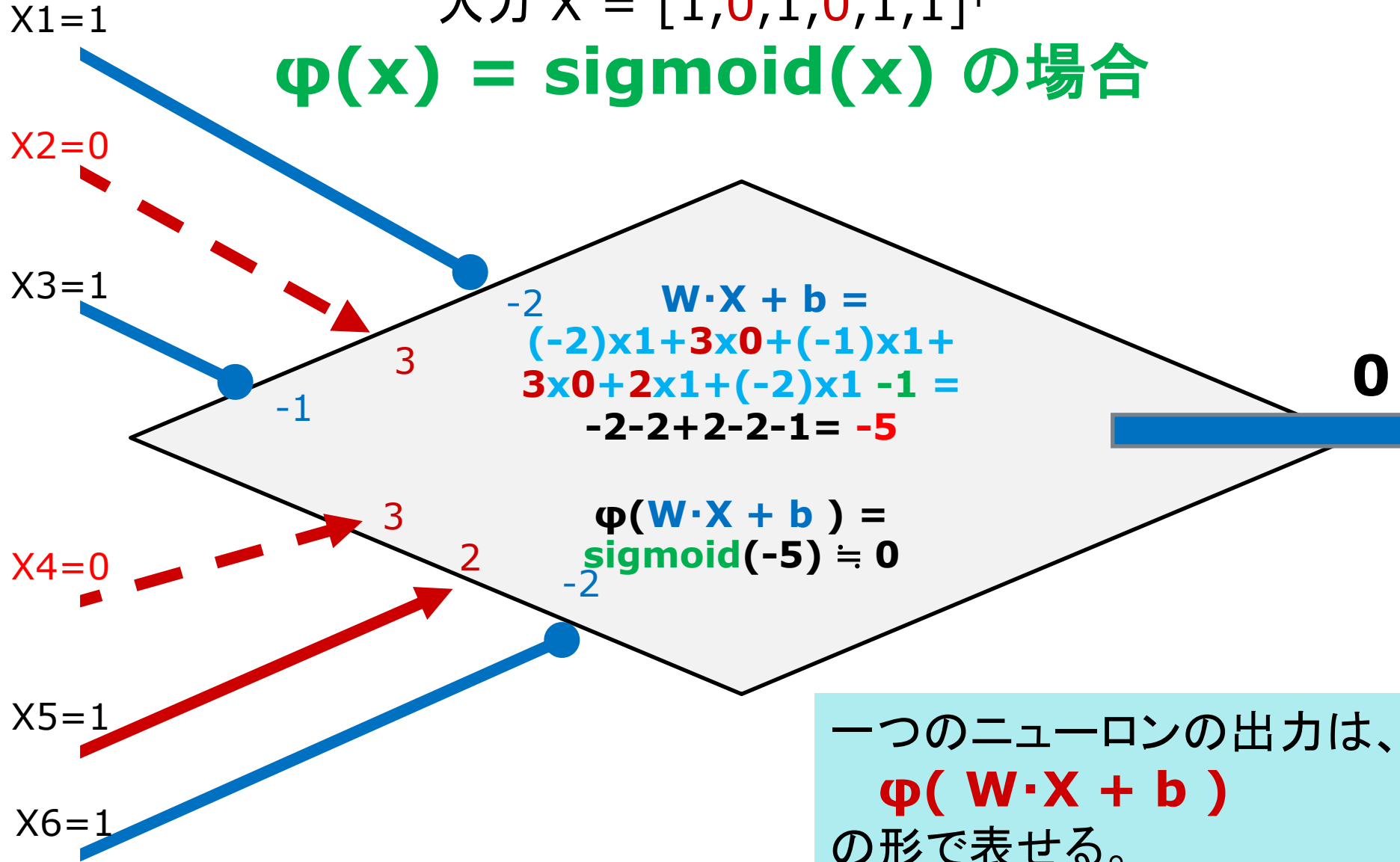
tanh 関数

- sigmoidの仲間だが、ちょうどそれを1/2ほど、下に移動した形である。**-1と+1との間の値を出力する**。tanh以外にも、同じ性質を持つ、様々な関数が提案されている。



重み $W = [-2, 3, -1, 3, 2, -2]$ 、バイアス $b = -1$
入力 $X = [1, 0, 1, 0, 1, 1]^T$

$\phi(x) = \text{sigmoid}(x)$ の場合



一つのニューロンの出力は、
 $\phi(W \cdot X + b)$
の形で表せる。

重み $W = [-2, 3, -1, 3, 2, -2]$ 、バイアス $b = -1$

入力 $X = [1, 1, 0, 1, 0, 1]^T$

$\phi(x) = \text{ReLU}(x)$ の場合

$X_1 = 1$

$X_2 = 1$

$X_3 = 0$

$X_4 = 1$

$X_5 = 0$

$X_6 = 1$

$$\begin{aligned} W \cdot X + b &= \\ (-2)x_1 + 3x_1 + (-1)x_0 + \\ 3x_1 + 2x_0 + (-2)x_1 - 1 &= \\ -2 + 3 + 3 - 2 &= 2 \end{aligned}$$

$$\begin{aligned} \phi(W \cdot X + b) &= \\ \text{ReLU}(2) &= 2 \end{aligned}$$

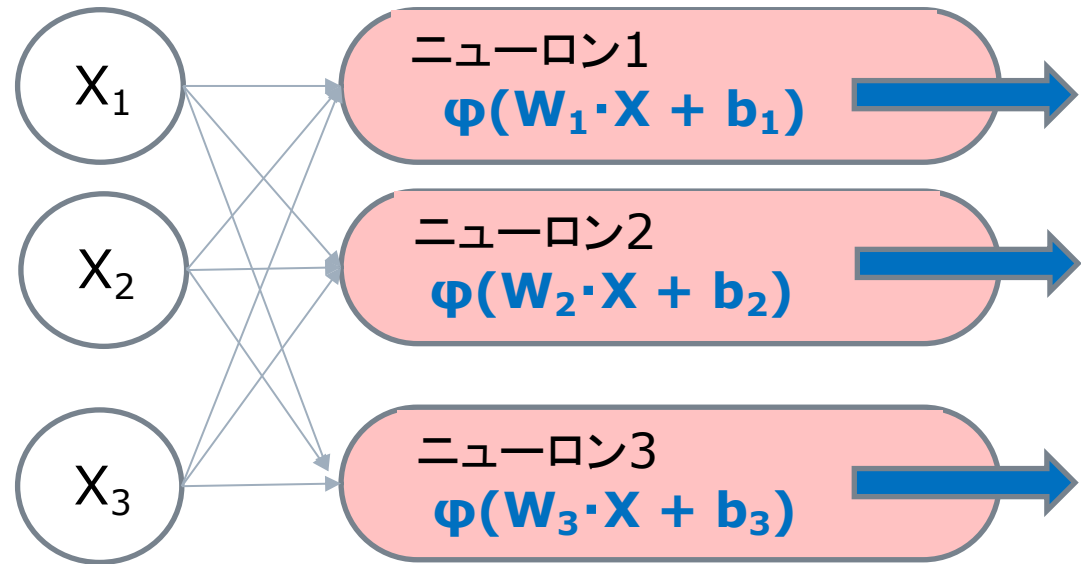
一つのニューロンの出力は、
 $\phi(W \cdot X + b)$
の形で表せる。

入力 $X=[X_1, X_2, X_3]^T$

重み $W_1=[W_{11}, W_{12}, W_{13}]$
バイアス b_1

重み $W_2=[W_{21}, W_{22}, W_{23}]$
バイアス b_2

重み $W_3=[W_{31}, W_{32}, W_{33}]$
バイアス b_3



$$\Rightarrow \begin{bmatrix} \varphi(W_{11}X_1 + W_{12}X_2 + W_{13}X_3 + b_1) \\ \varphi(W_{21}X_1 + W_{22}X_2 + W_{23}X_3 + b_2) \\ \varphi(W_{31}X_1 + W_{32}X_2 + W_{33}X_3 + b_3) \end{bmatrix} = \varphi \left\{ \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right\}$$

複数のニューロンの出力も、 $\varphi(W \cdot X + b)$ の形で表せる。

一つのニューロンの出力も、
複数のニューロンの出力も、

$\phi(W \cdot X + b)$ の形で表せる。



この式は、ニューラルネットワークで、
もっとも重要な式の一つである。



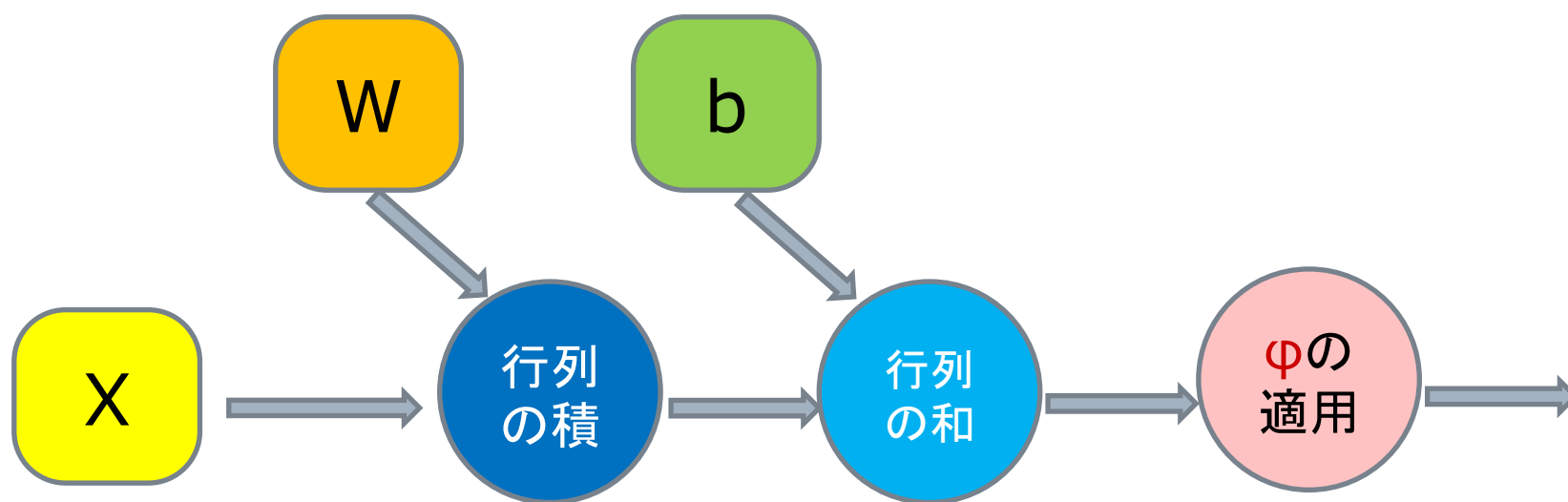
ニューラル・ネットワークを
グラフで表現する

複数のニューロンからなる一つの層を、 グラフで表す

ここから、TensorFlowを使った、ニューラル・ネットワークの実装のスタイルを紹介する。

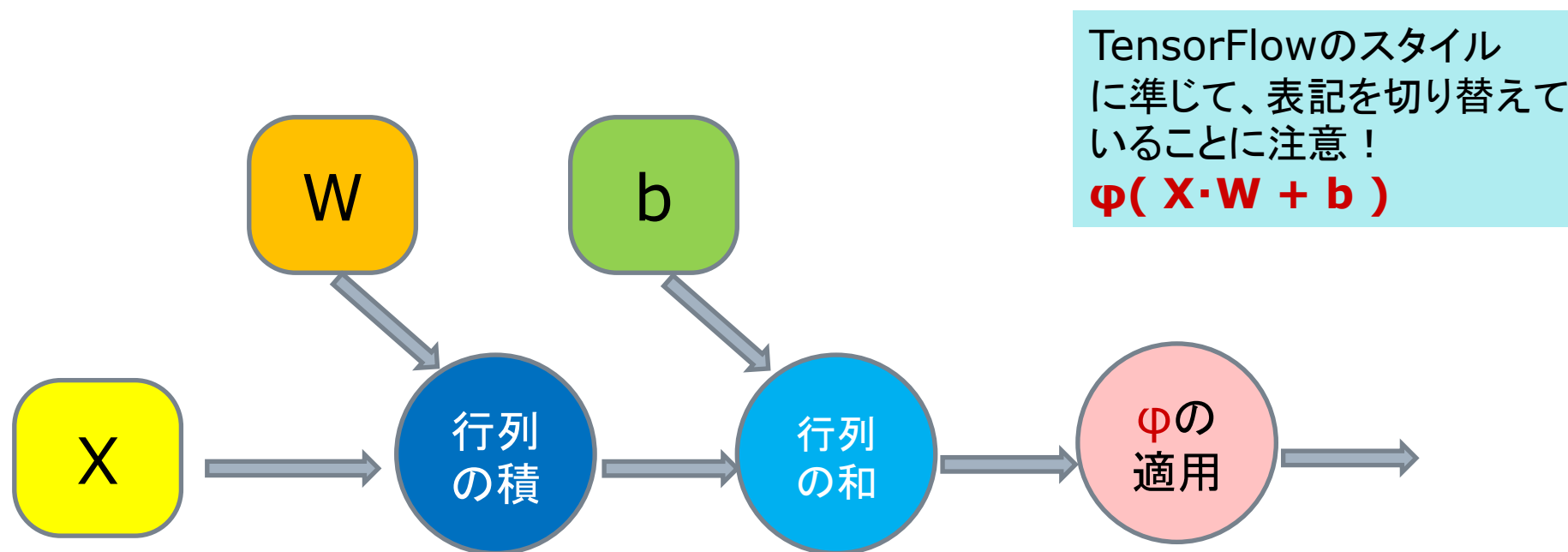
「グラフ」「演算ノード」「テンソル」というのが、基本的なコンセプトとなる。

複数のニューロンからなる一つの層の出力の計算、
 $\phi(W \cdot X + b)$ を、次のような図形で表わすことにしよう。



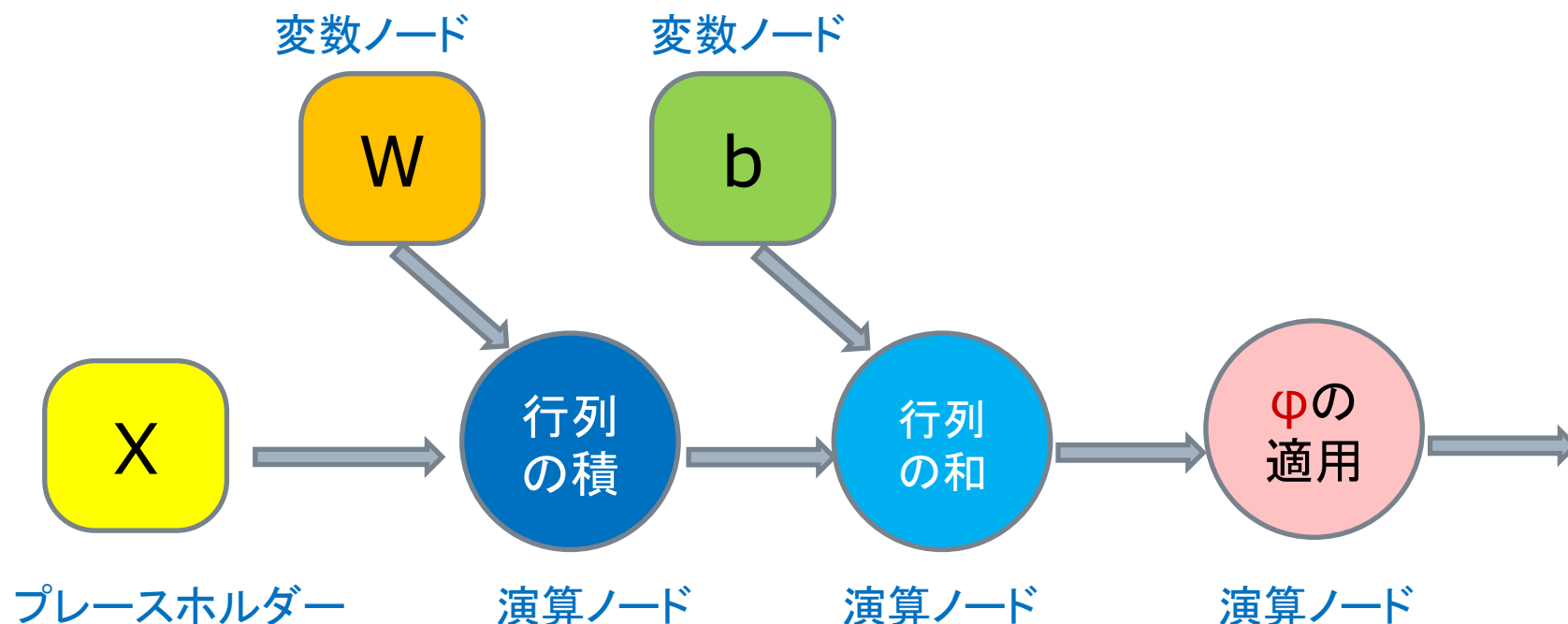
先の例で言うと、
 X は、 $[X_1, X_2, X_3]^T$ の列ベクトルで、
 W は 3×3 の行列、
 b は、 $[b_1, b_2, b_3]^T$ の列ベクトルである。

複数のニューロンからなる一つの層の出力の計算、
 $\phi(X \cdot W + b)$ を、次のような図形で表わすことにしよう。



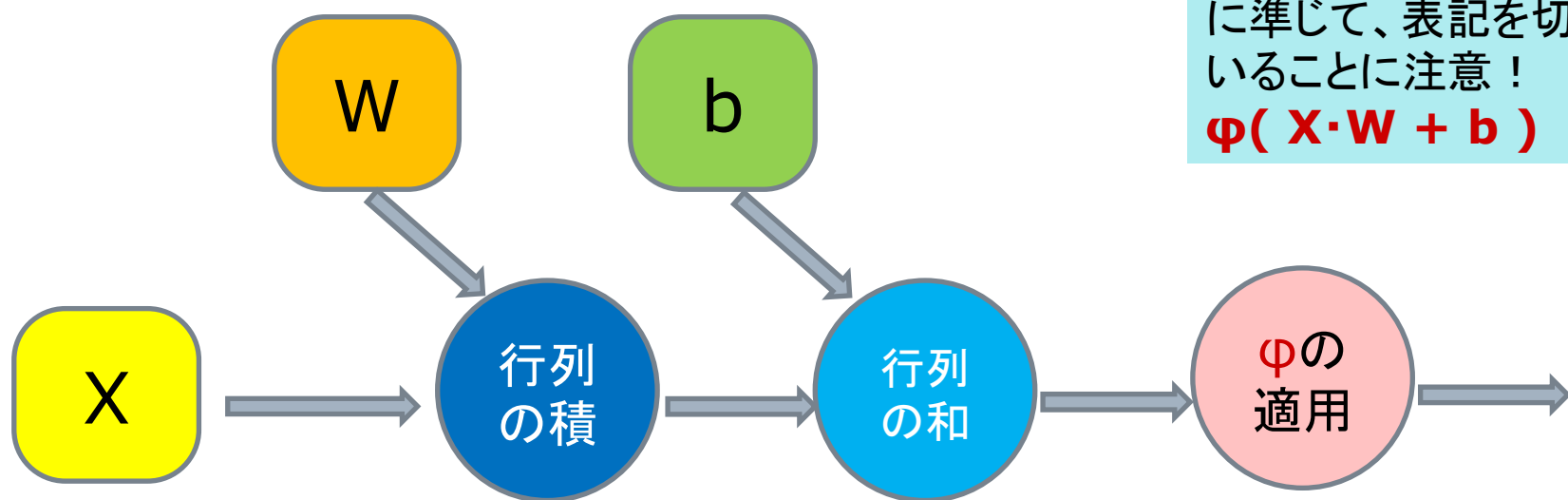
先の例で言うと、
 X は、 $[X_1, X_2, X_3]$ の行ベクトルで、
 W は 3×3 の行列、
 b は、 $[b_1, b_2, b_3]$ の行ベクトルである。

TensorFlowでは、こうした図形を「グラフ」と呼んでいる。
グラフは、「ノード」とノード同士を結ぶ「辺」からできている。



TensorFlowのグラフのノードは、基本的には、演算を行う「演算ノード」である。様々な演算が用意されている。演算を行わない「変数ノード」「プレースホルダー・ノード」も存在する。

TensorFlowのグラフが有用なのは、一つの同じグラフで、様々な複雑なニューラル・ネットワークを表現できるからである。
(もちろん、 X, W, b のタイプはことなるのだが)

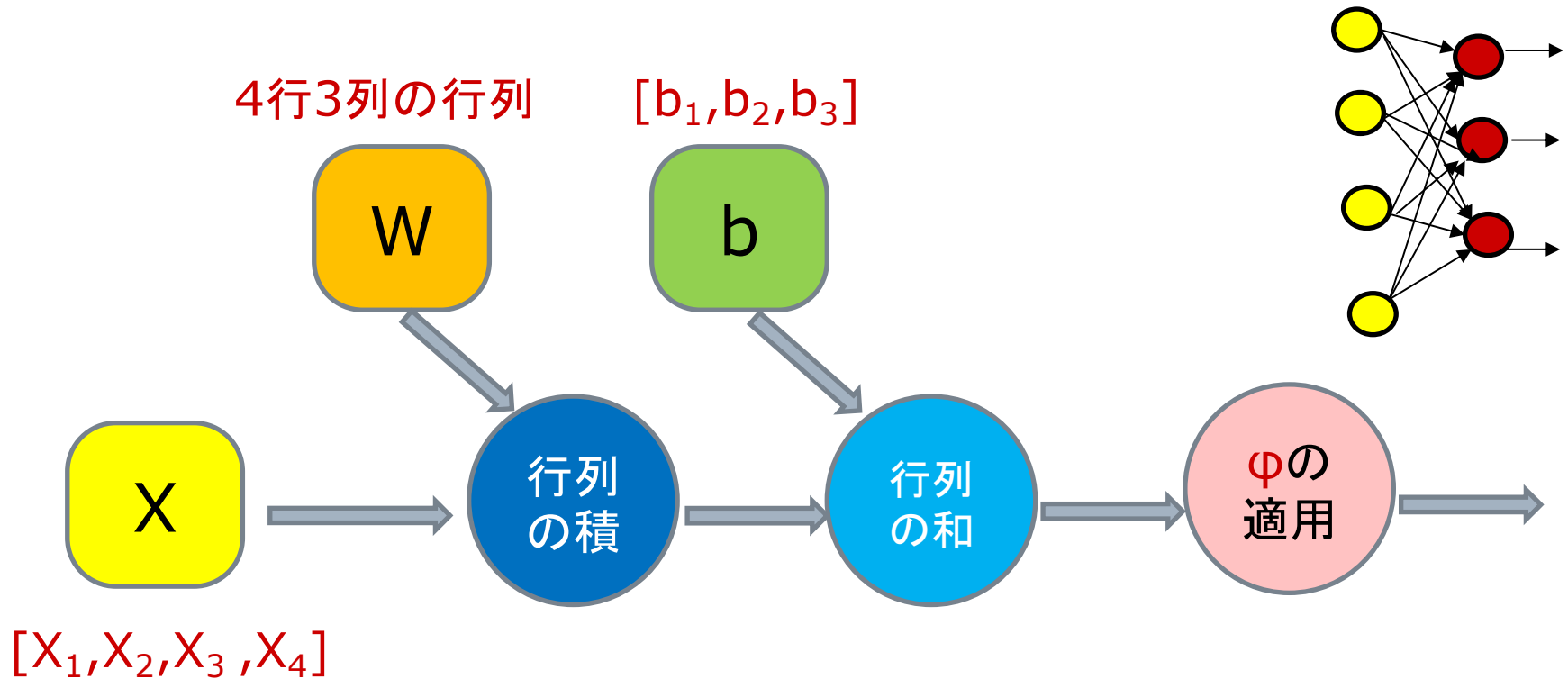


TensorFlowのスタイルに準じて、表記を切り替えていることに注意！
 $\phi(X \cdot W + b)$

もっとも、そのことは、基本的には、このグラフの場合には、すでに見た **$\phi(X \cdot W + b)$** という数学的な抽象化に負っている。ただ、グラフの方が具体的に処理をイメージしやすい。

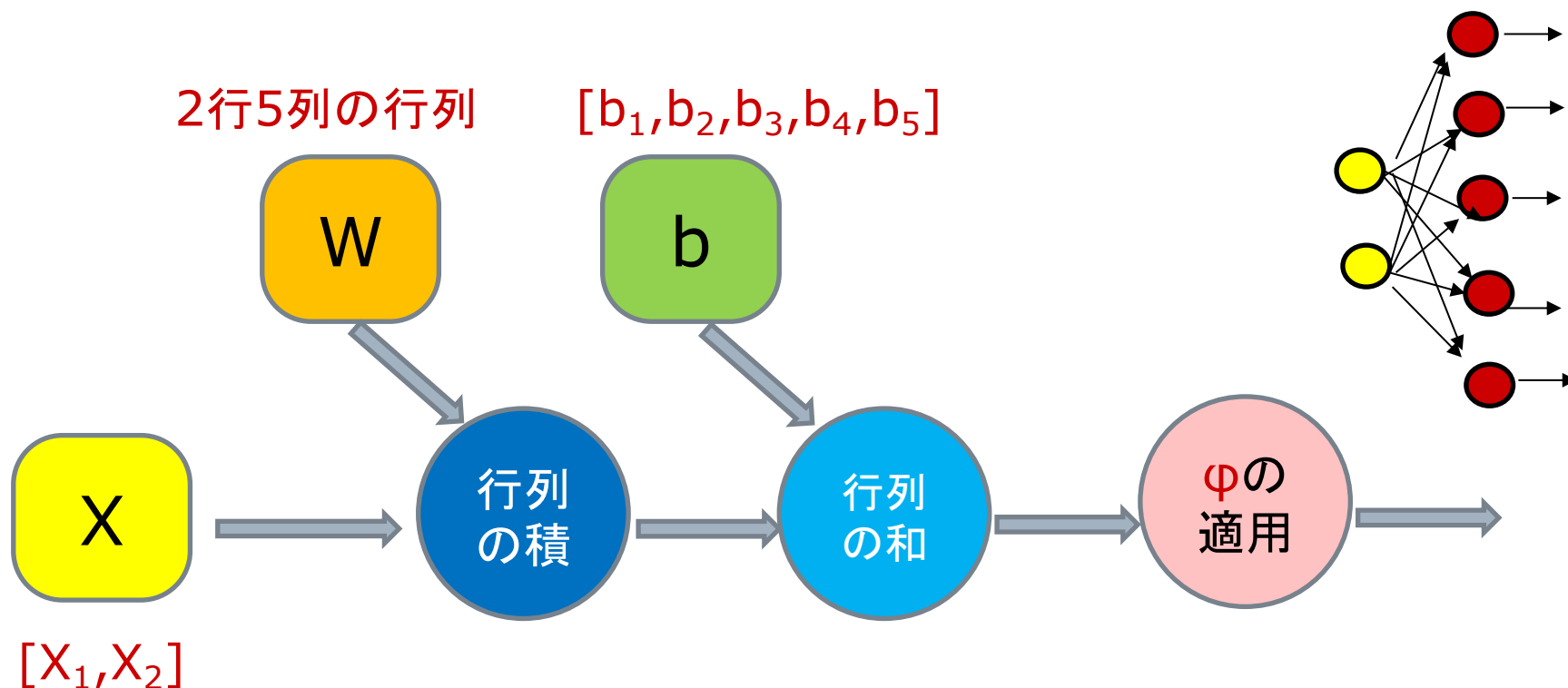
いくつかの例で考えてみよう。

4つの入力を受け取り、3つの出力を返す、3つのニューロン
からなる「層」を考えてみよう。これも、このグラフで表現できる。



ある「層」の出力の数は、その層に含まれるニューロンの数に
等しいのは、当然である。

2つの入力を受け取り、5つの出力を返す、5つのニューロンからなる「層」を考えてみよう。これも、このグラフで表現できる。



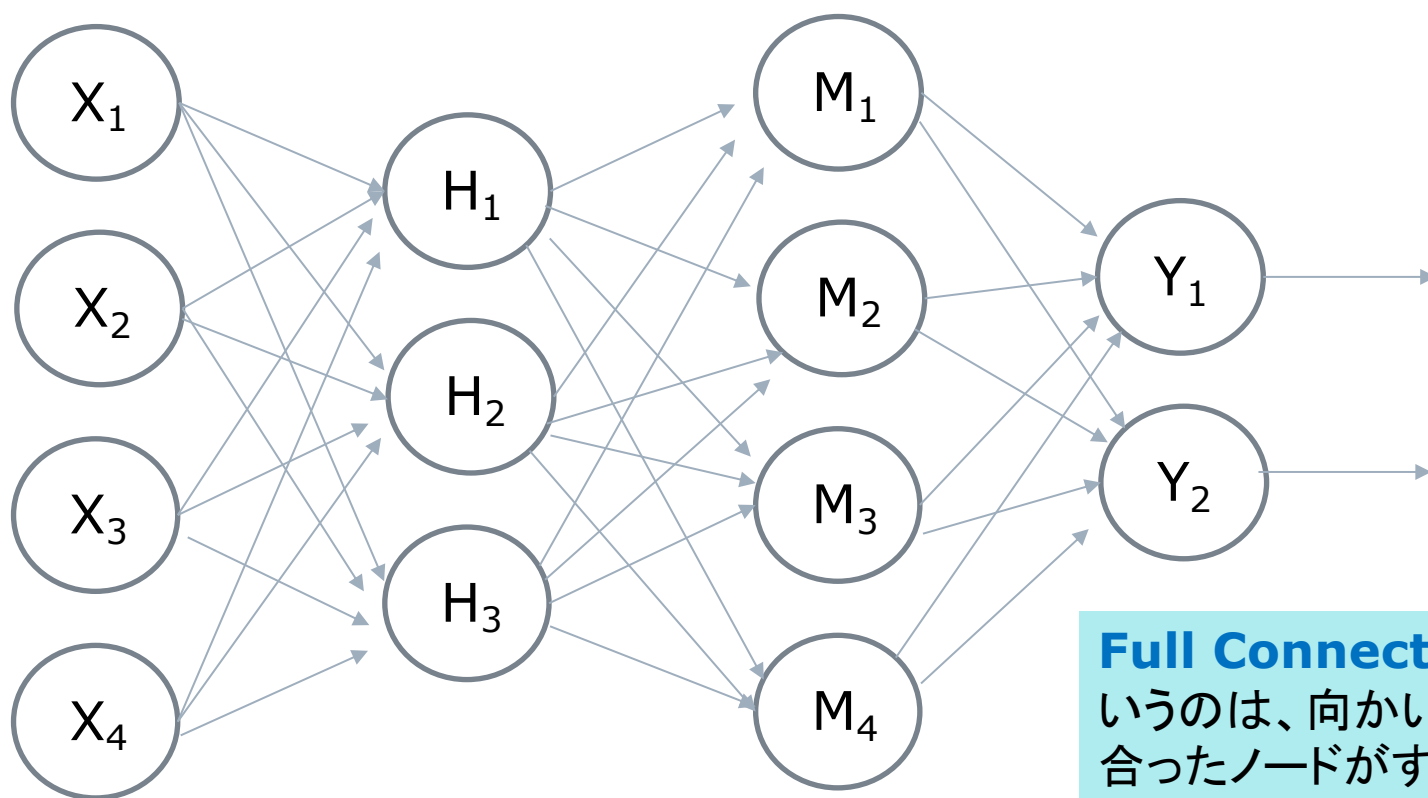
一般に、 n 個の入力を受け取り、 m 個の出力を返すニューロンの「層」では、 X は n 次元の行ベクトル、 W は n 行 m 列の行列、 b は m 次元の行ベクトルである。

複数の層からなるニューラル・ネットワークを、グラフで表す

もちろん、通常の表現でも、ニューラル・ネットワークはグラフとして表現される。ただ、TensorFlowのグラフは、それをさらに単純化する。

そうした単純化されたグラフ表現は、TensorFlowの大きな能力である。そのことと、グラフを流れる「テンソル」というデータの捉え方とは、結びついている。

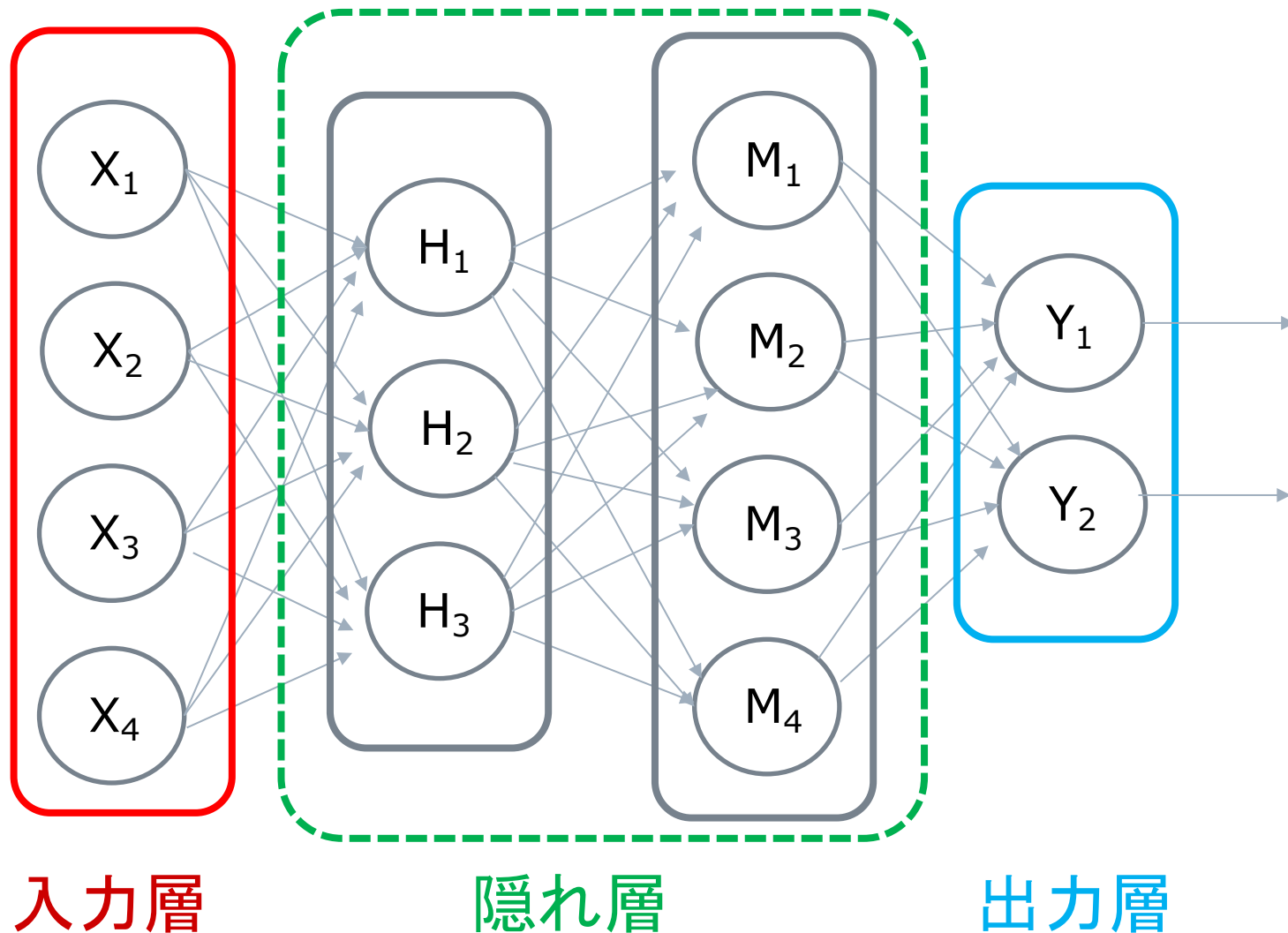
Full Connectなニューラル・ネットワークの 旧来のグラフでの表現



Full Connectと
いうのは、向かい
合ったノードがす
べて接続されている
ということ

入力層・隠れ層・出力層

旧来のグラフでの表現

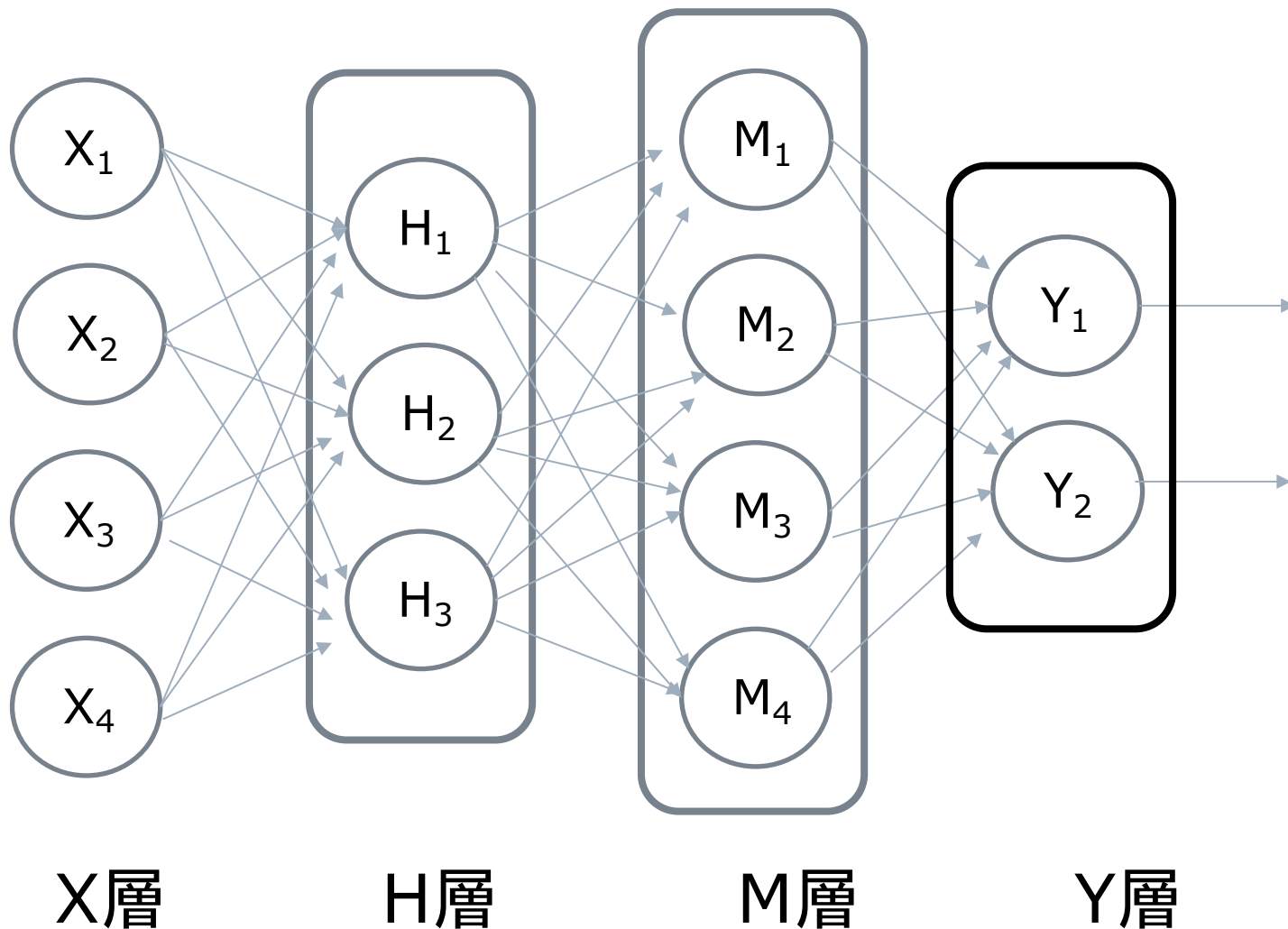


旧来のグラフ

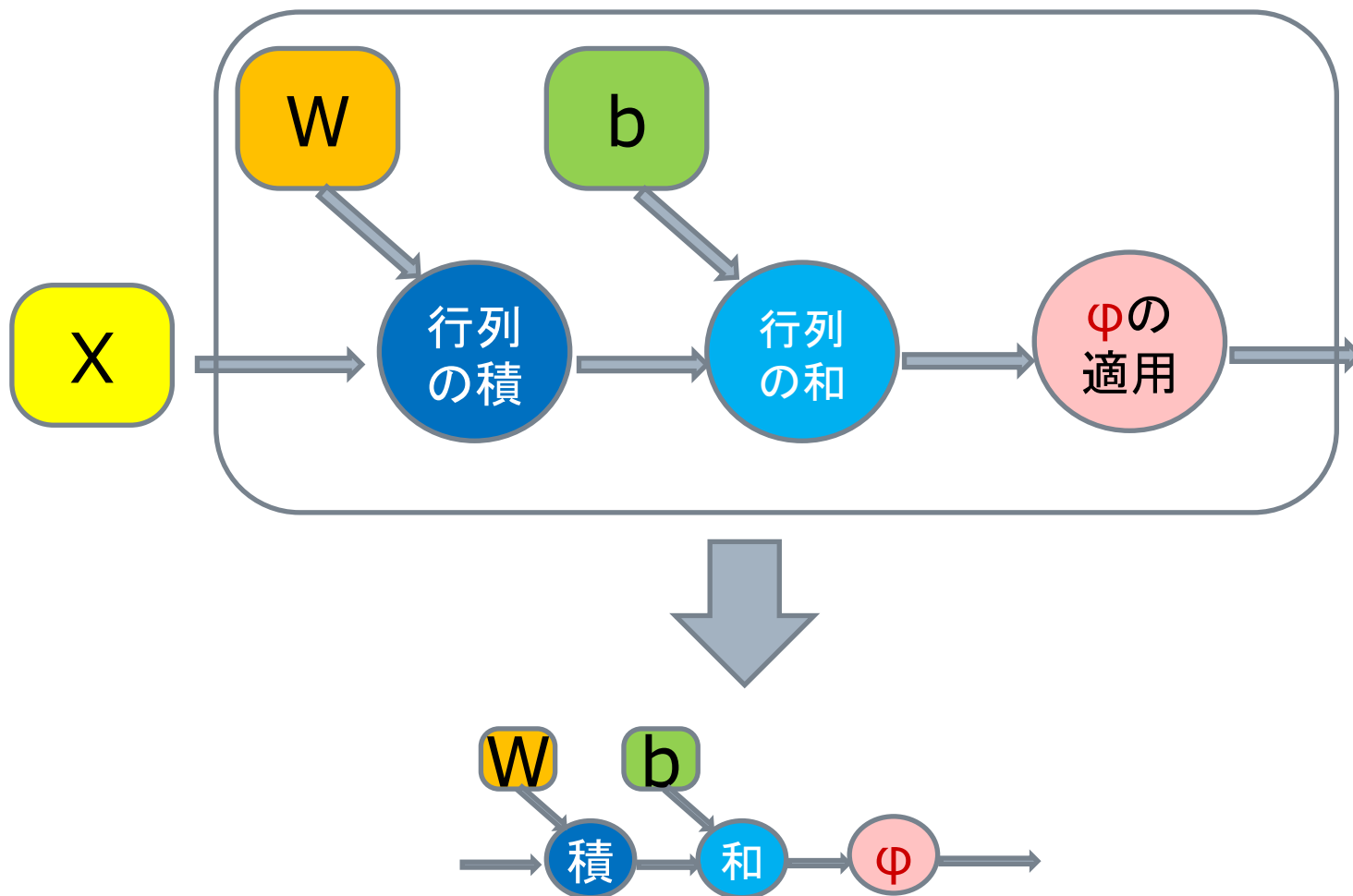
H層の
重み W_H
バイアス b_H
活性化 ϕ_H

M層の
重み W_M
バイアス b_M
活性化 ϕ_M

Y層の
重み W_Y
バイアス b_Y
活性化 ϕ_Y



一つの「層」のグラフでの表現

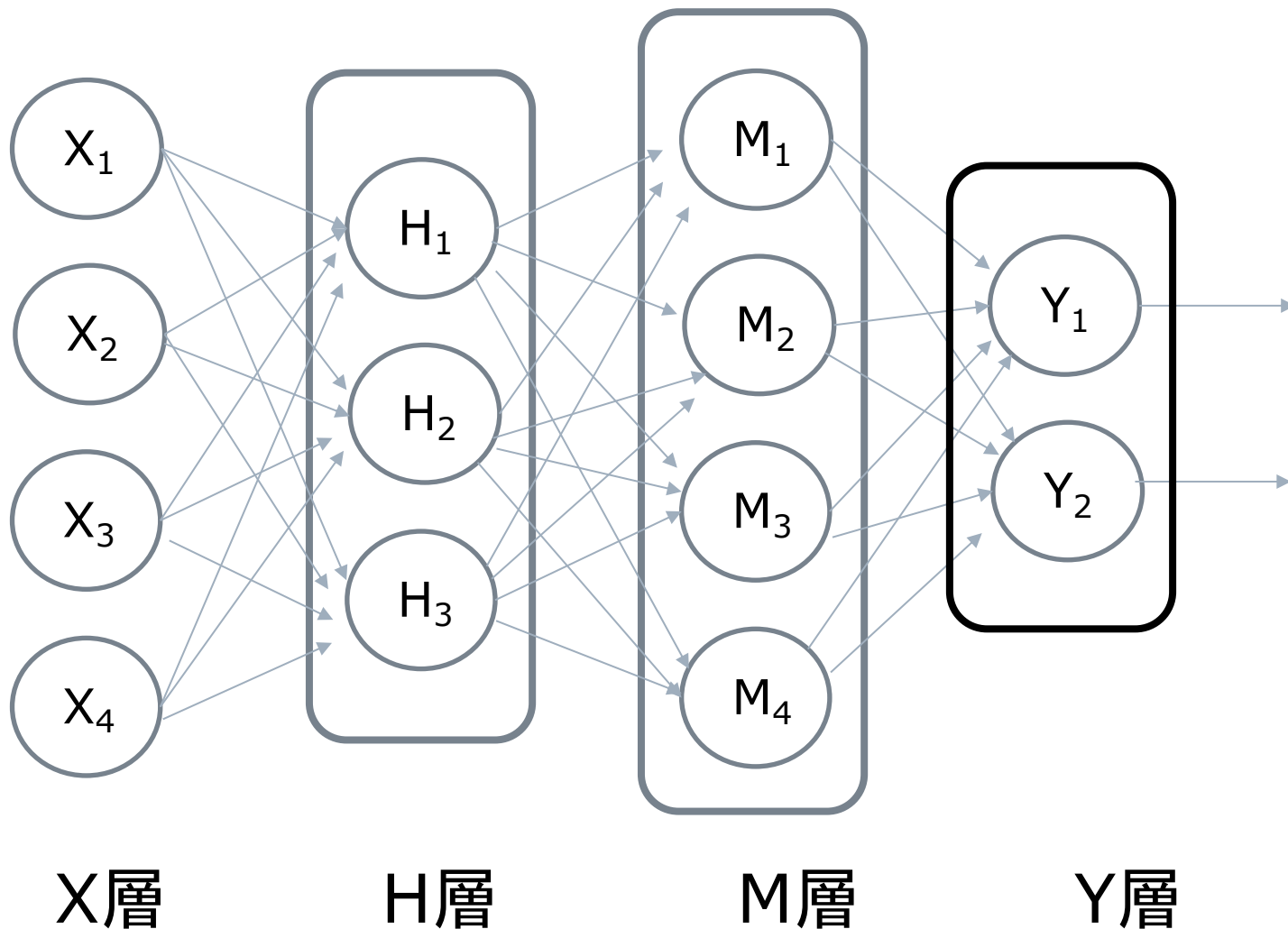


旧来のグラフ

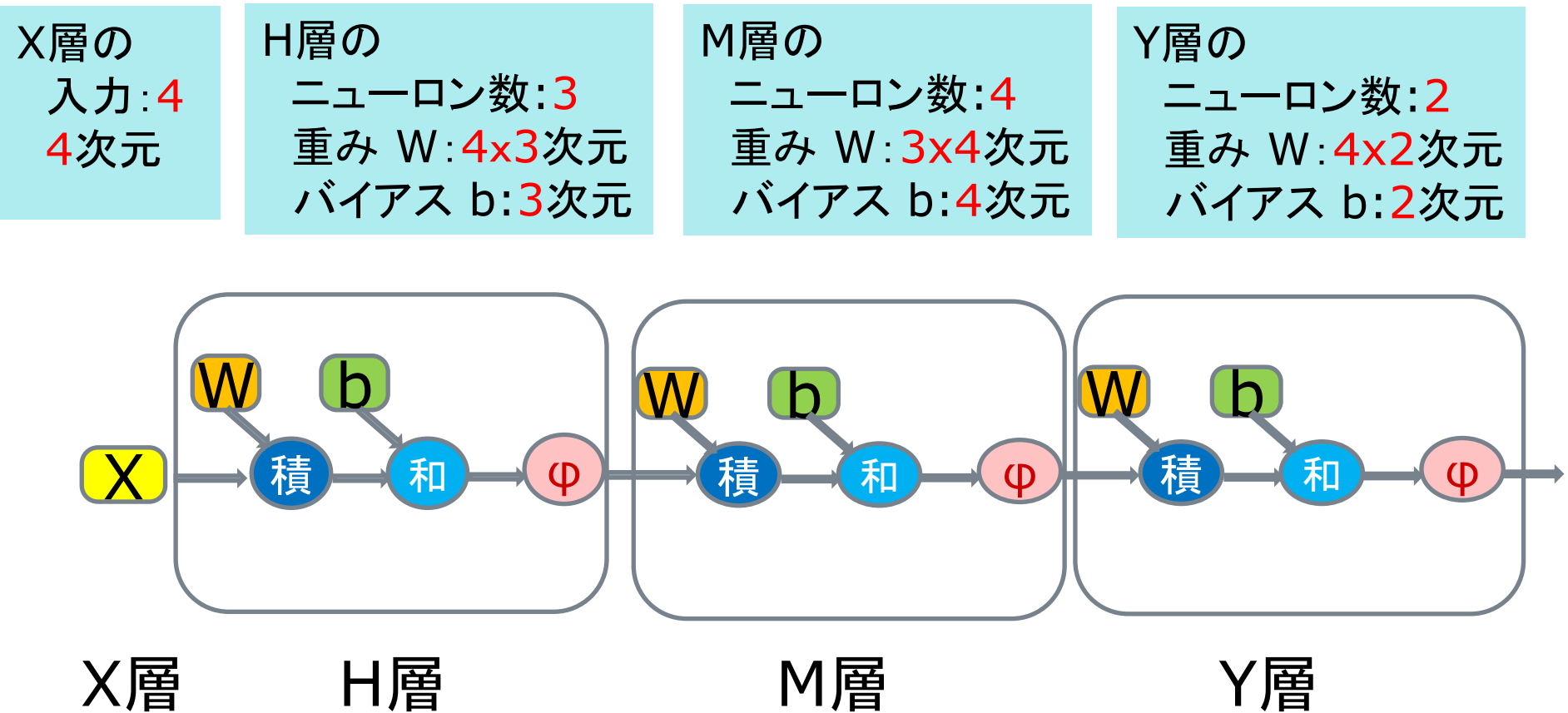
H層の
重み W_H
バイアス b_H
活性化 ϕ_H

M層の
重み W_M
バイアス b_M
活性化 ϕ_M

Y層の
重み W_Y
バイアス b_Y
活性化 ϕ_Y



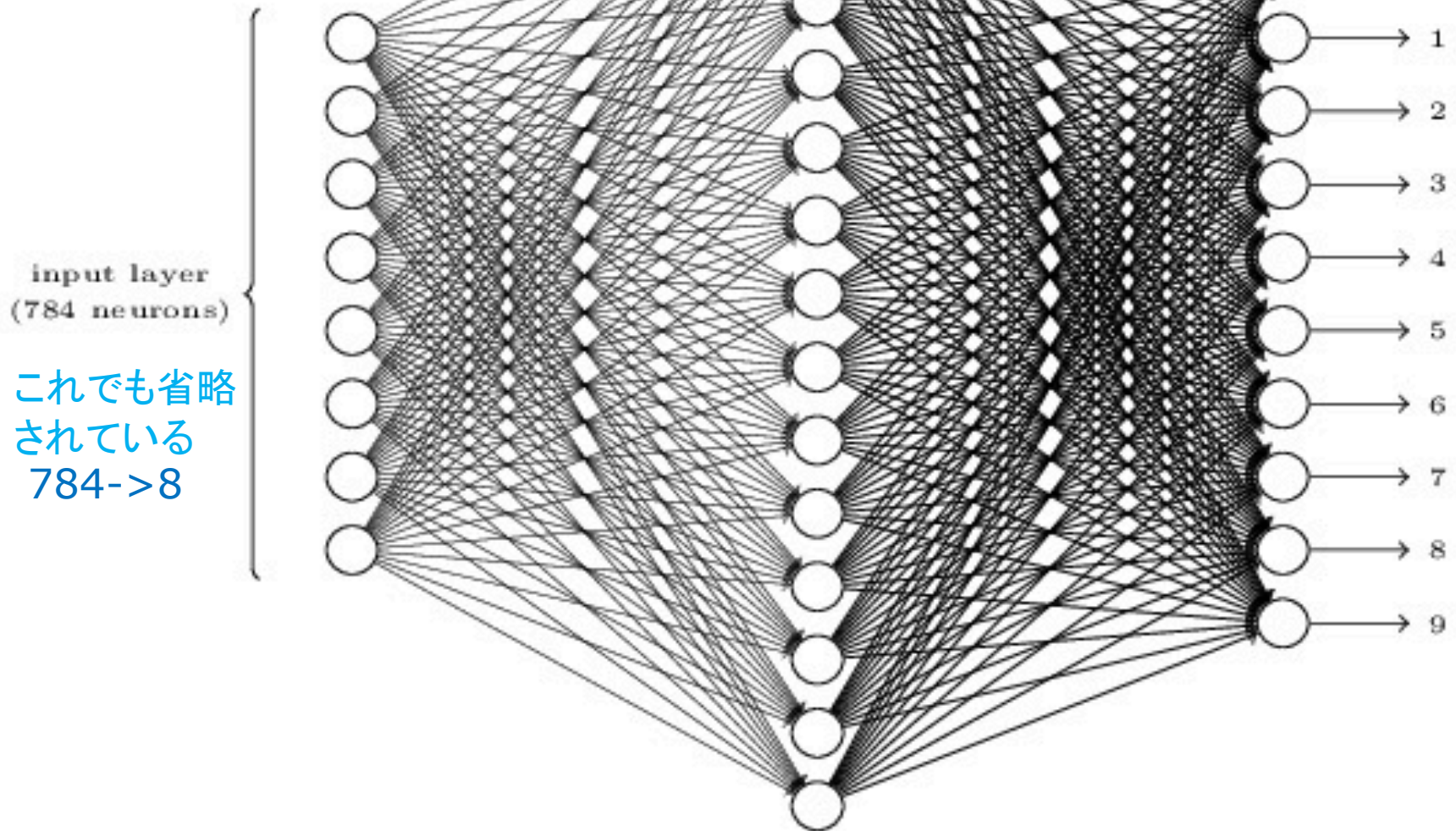
TensorFlowのグラフ



こっちのグラフの方が、ずっとわかりやすい！

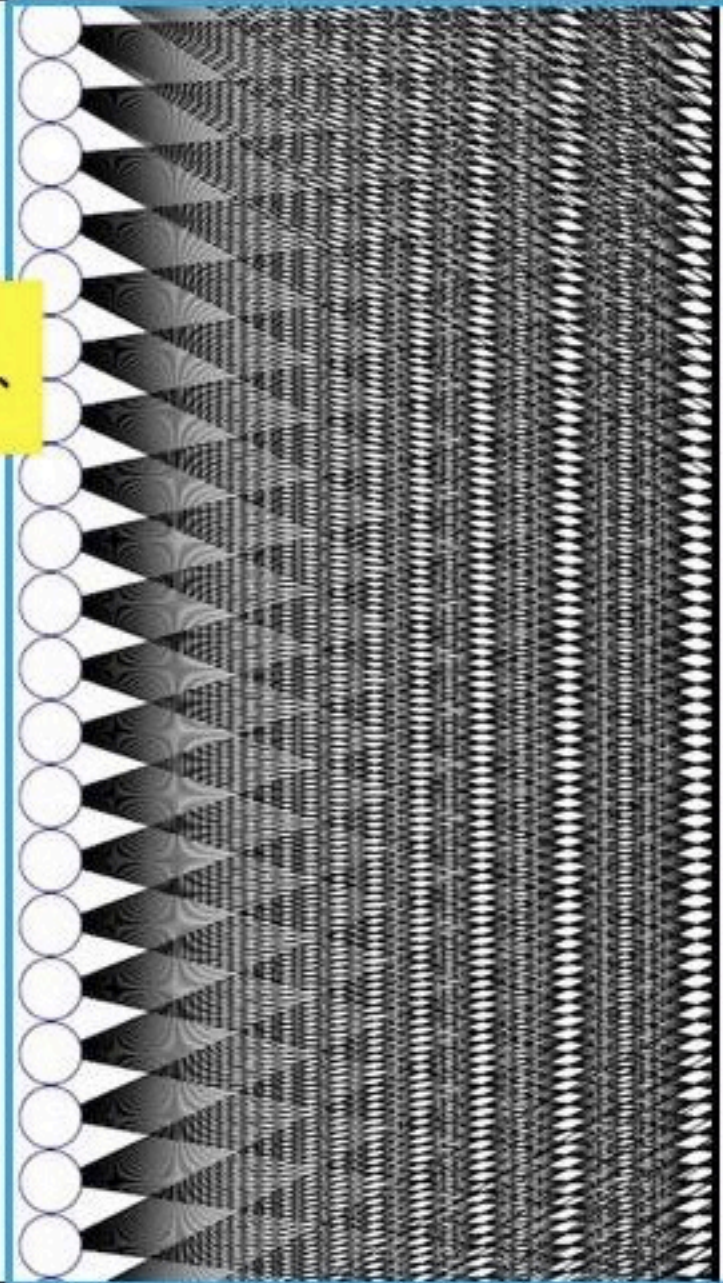
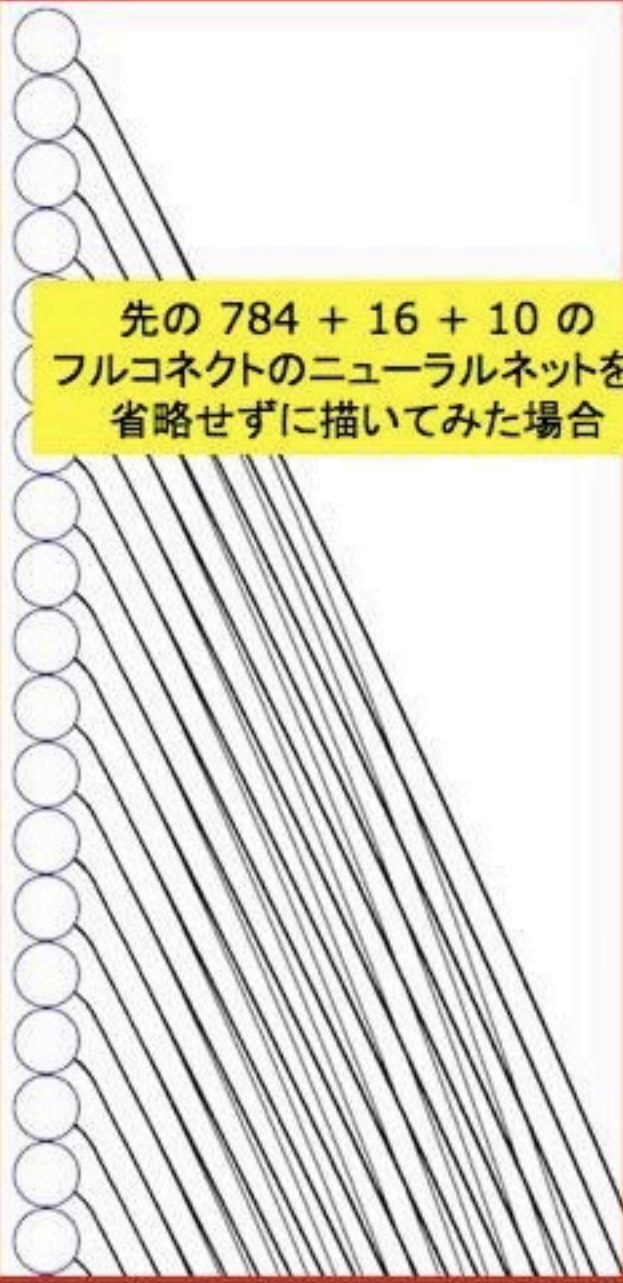
ニューラル・ネットワークの例

旧来のグラフ





先の $784 + 16 + 10$ の
フルコネクのニューラルネットを、
省略せずに描いてみた場合

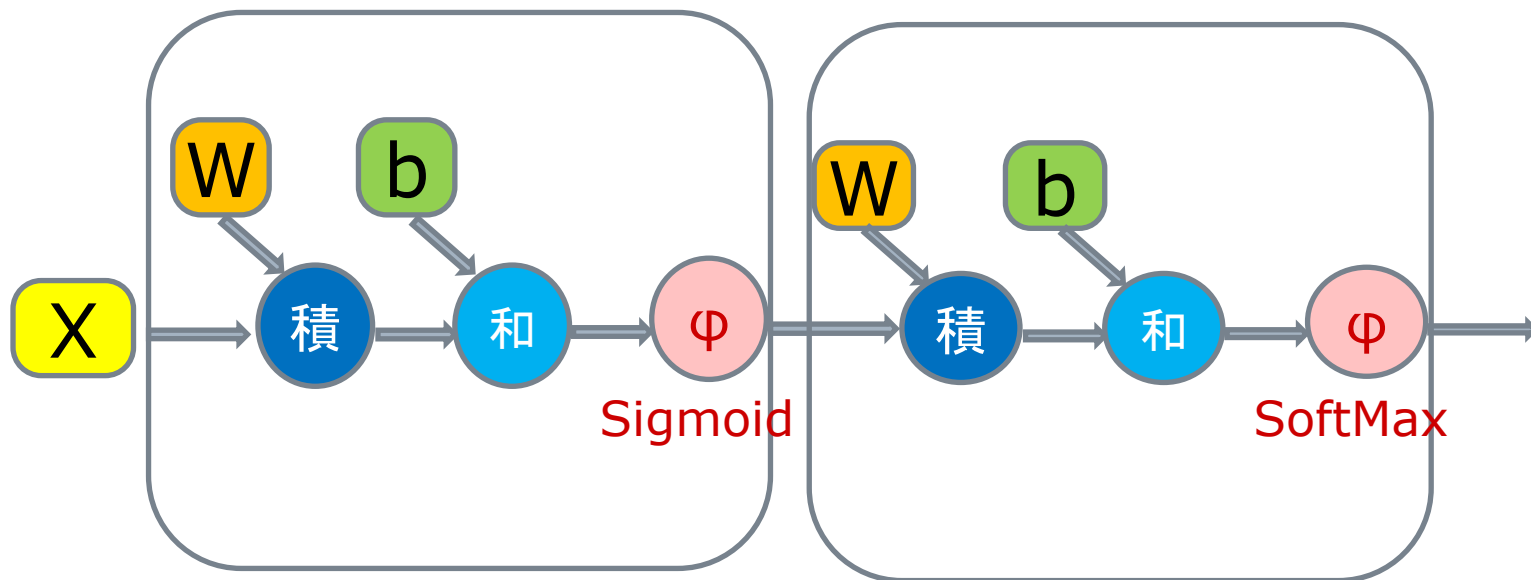


TensorFlowのグラフ

隠れ層の
入力: 784
784次元

隠れ層の
ニューロン数: 15
重み W_H : 784x15次元
バイアス b_H : 15次元

出力層の
ニューロン数: 10
重み W_O : 15x10次元
バイアス b_O : 10次元



入力層

隠れ層

出力層

こっちのグラフの方が、ずっとわかりやすい！

グラフの例 (学習用)

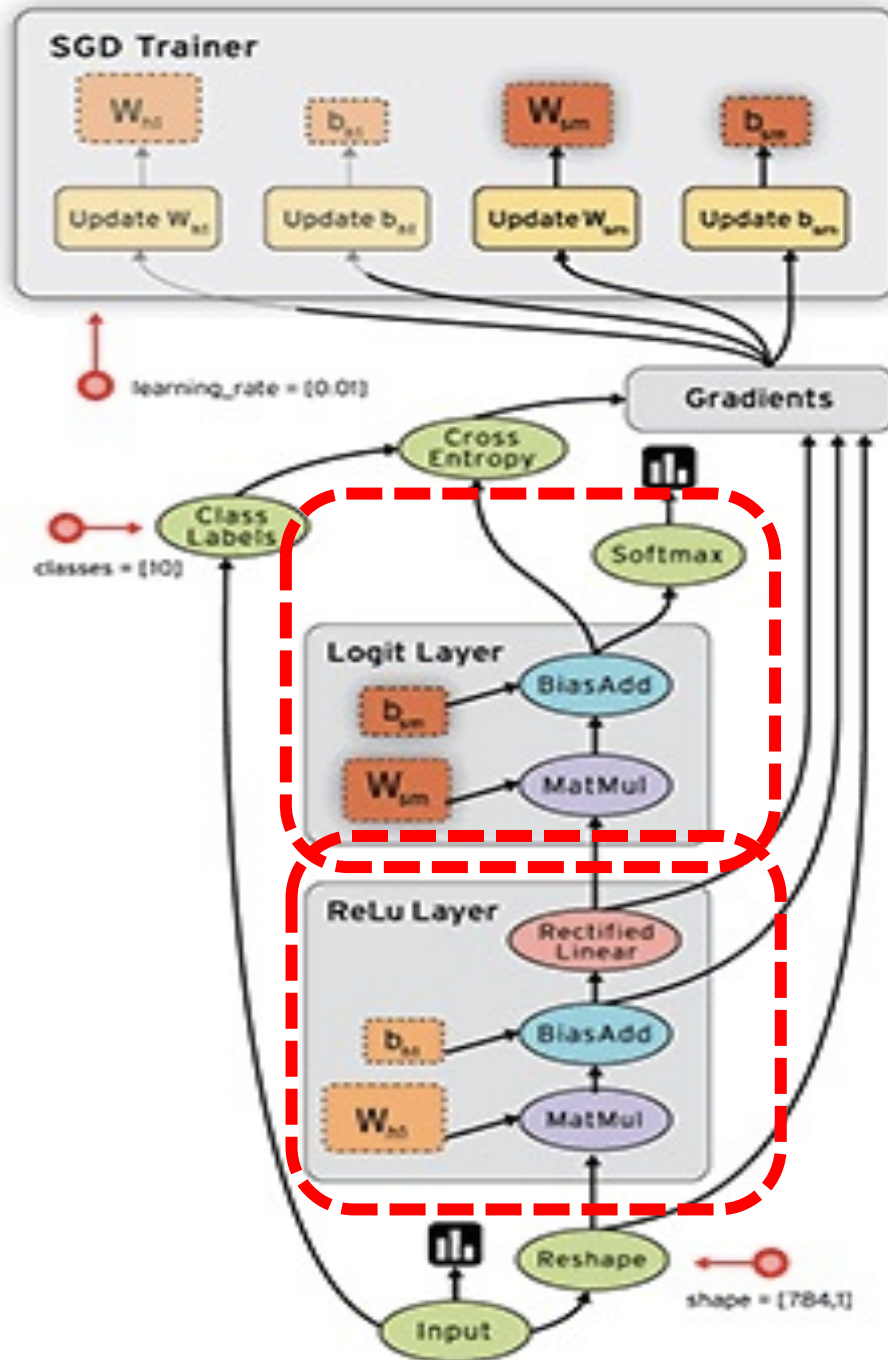
<https://www.tensorflow.org/>

Logit Layer, ReLu Layerを見れば、先に見た「層」のパターンが現れているのがわかる。

演算ノードの、MatMulは行列の積、BiasAddはバイアスの加算である。

Logit LayerのActivatorは、Softmaxで、ReLu LayerのActivatorは、ReLuである。

(このグラフ、GitHubで提供されているコードのグラフとは、少し、違っている。)

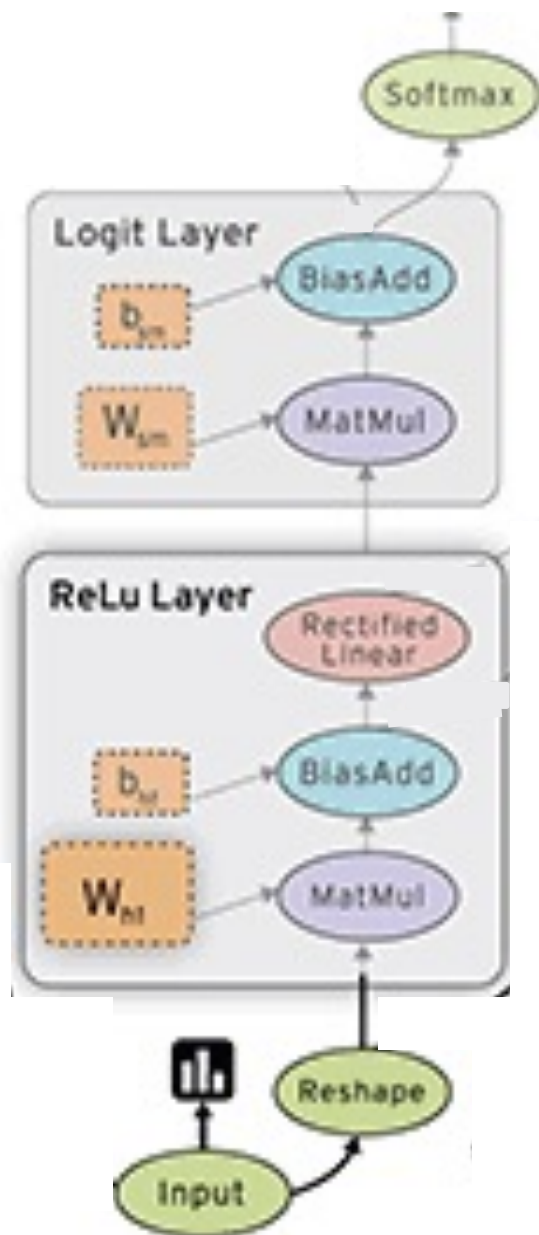


グラフの例

(学習済み)

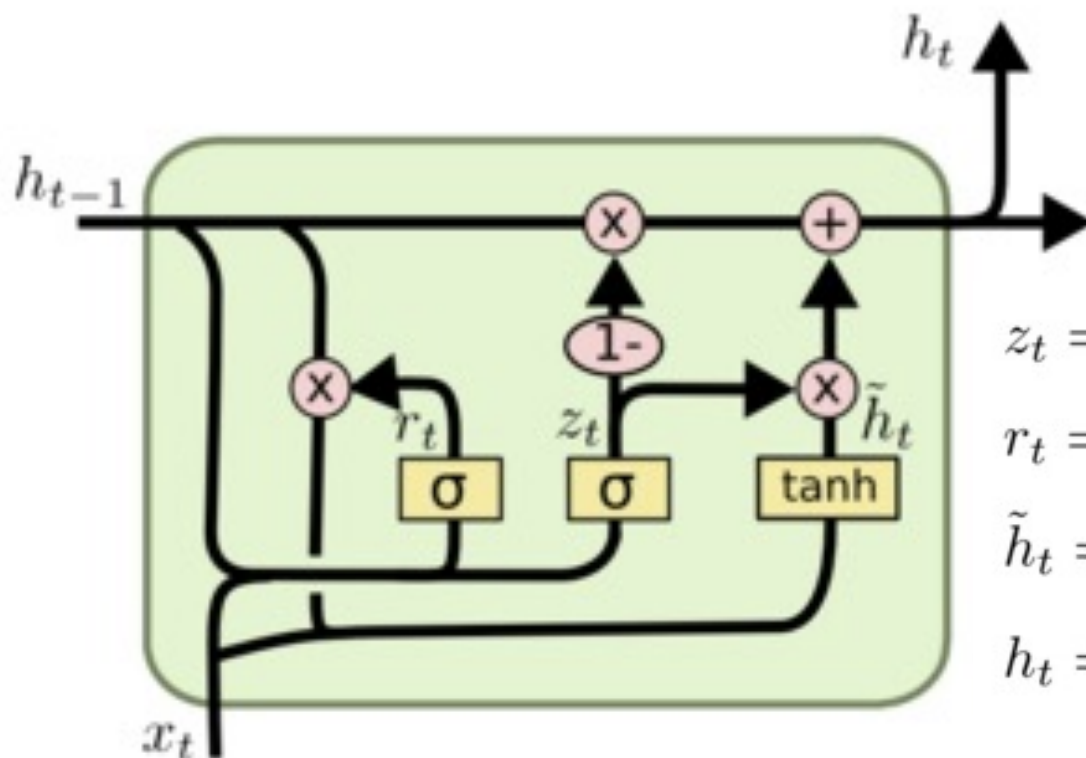
ニューラル・ネットワークでは、学習の過程と、学習済みの結果を利用する過程では、計算量には、大きな差がある。

学習に持ちいたグラフの一部を再利用して、学習済みのデータ（基本的には、学習された各層の重みとバイアスである）を利用すれば、**スマホでも機械学習の成果を利用できる。**
TensorFlowは、それを可能にする



少し複雑なグラフの例 (これは、LSTMのセル)
 σ (sigmoidのこと), $\tanh$ は、活性化関数

次の式をグラフで
追ってみるといい。



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Chris Olah "Understanding LSTM Networks"

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>



ニューラル・ネットワークの グラフ表現とモジュール構造

ニューラル・ネットワークの グラフ表現とモジュール構造

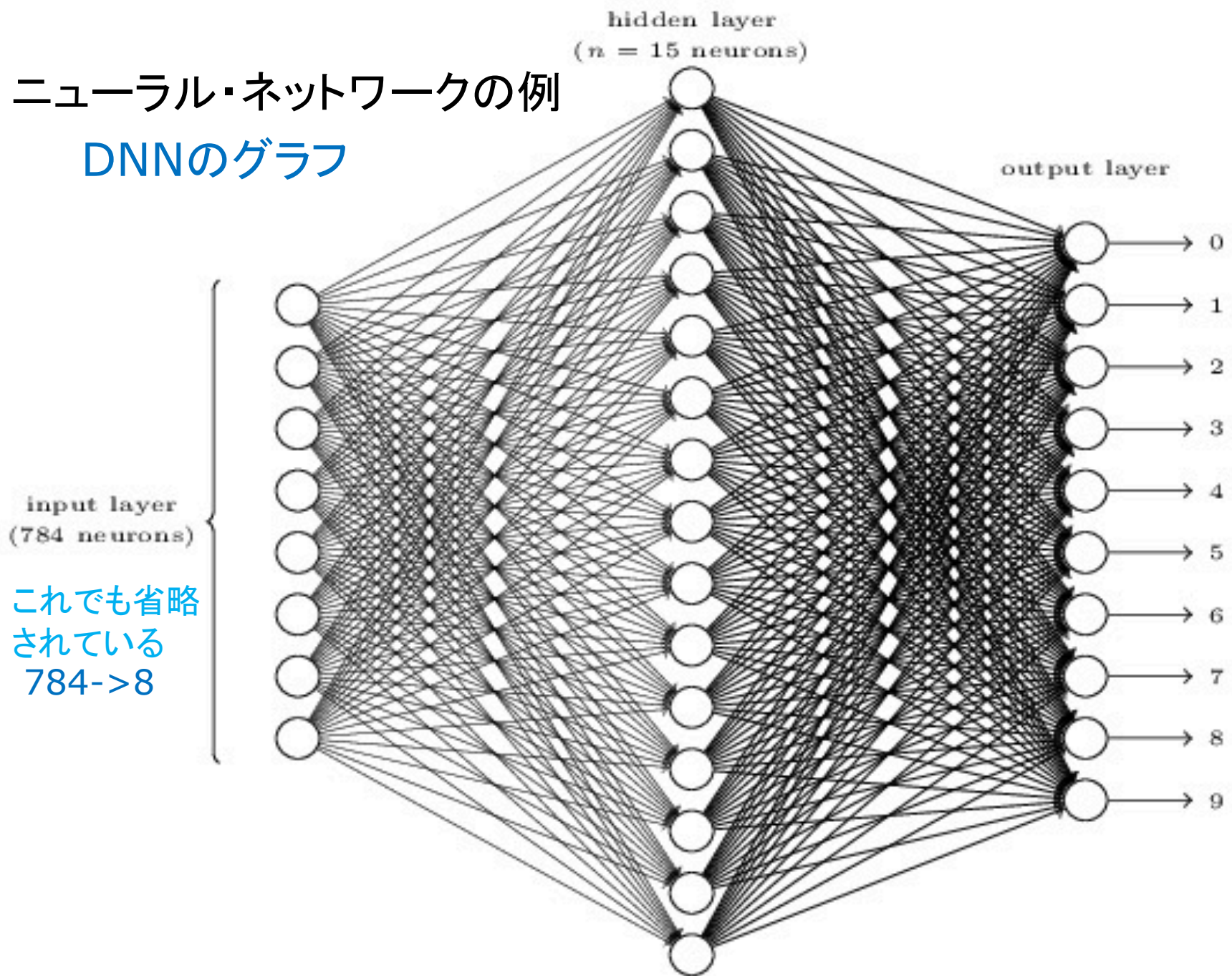
それぞれのフレームワークごとに表現のスタイルには違いがあるものの、重要なことは、それが表現しているニューラル・ネットワークの「グラフ」は、同一のものであるということである。それは、基本的なモジュールの組み合わせでできている。

フルコネクトDNNの グラフ表現とモジュール構造

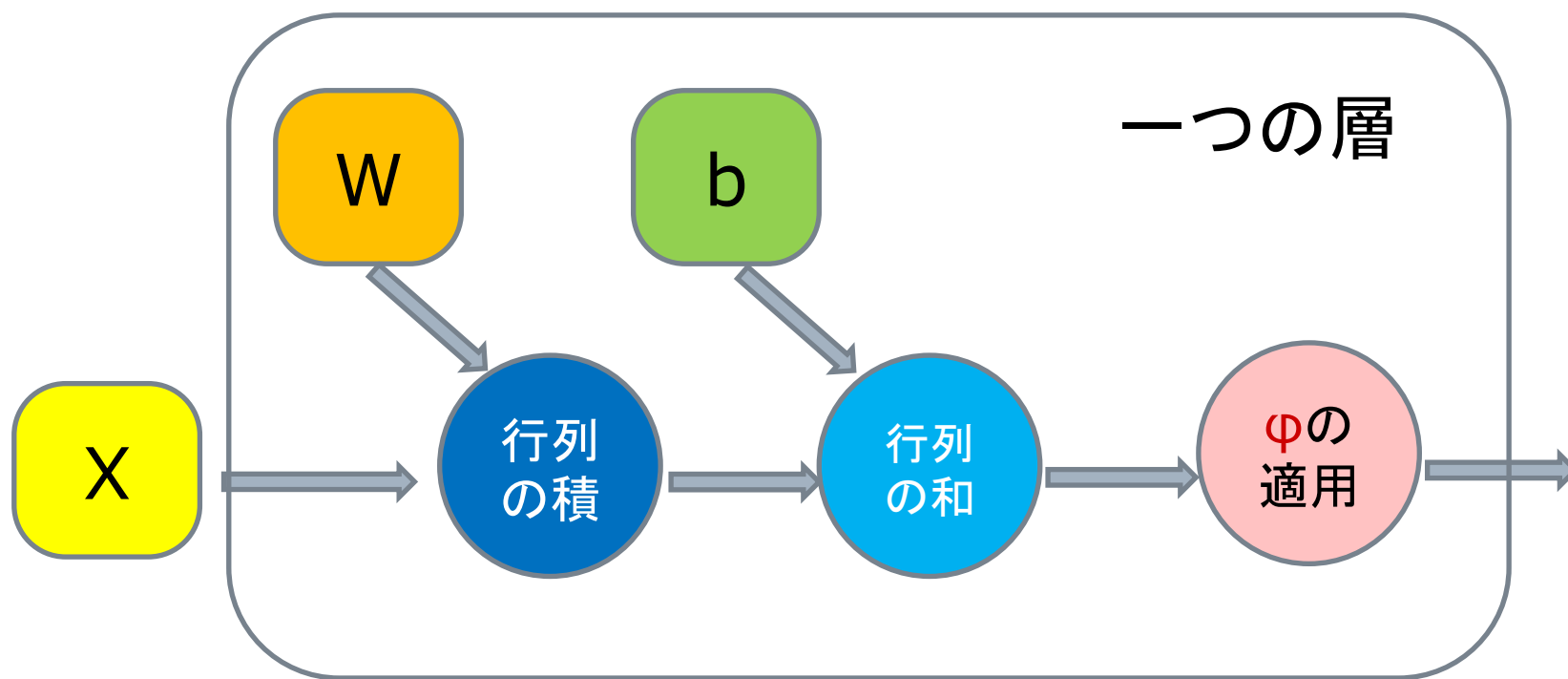
まずは、基本的な フルコネクトDNN (Deep Neural Network)のグラフ表現とモジュール構造を見ていこう

ニューラル・ネットワークの例

DNNのグラフ



TensorFlowでは、複数のニューロンからなる一つの層は、次のようなグラフで表現される。



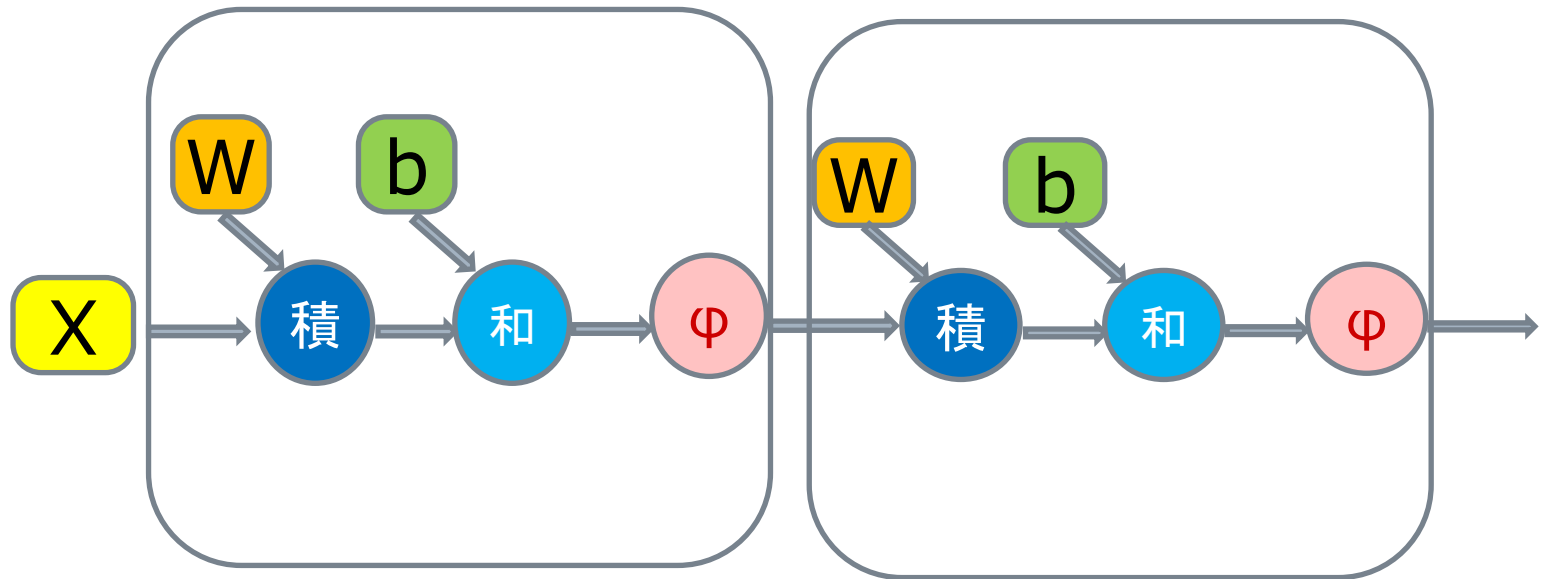
5つのノードがあるが、Wとbには、重みとバイアスの値が入り、残りの3つのノードは、演算を行うノードである。この層の働きは、 **$\phi(X \cdot W + b)$** で表現される。

TensorFlowのグラフ

隠れ層の
入力: 784
784次元

隠れ層の
ニューロン数: 15
重み W_H : 784×15次元
バイアス b_H : 15次元

出力層の
ニューロン数: 10
重み W_O : 15×10次元
バイアス b_O : 10次元



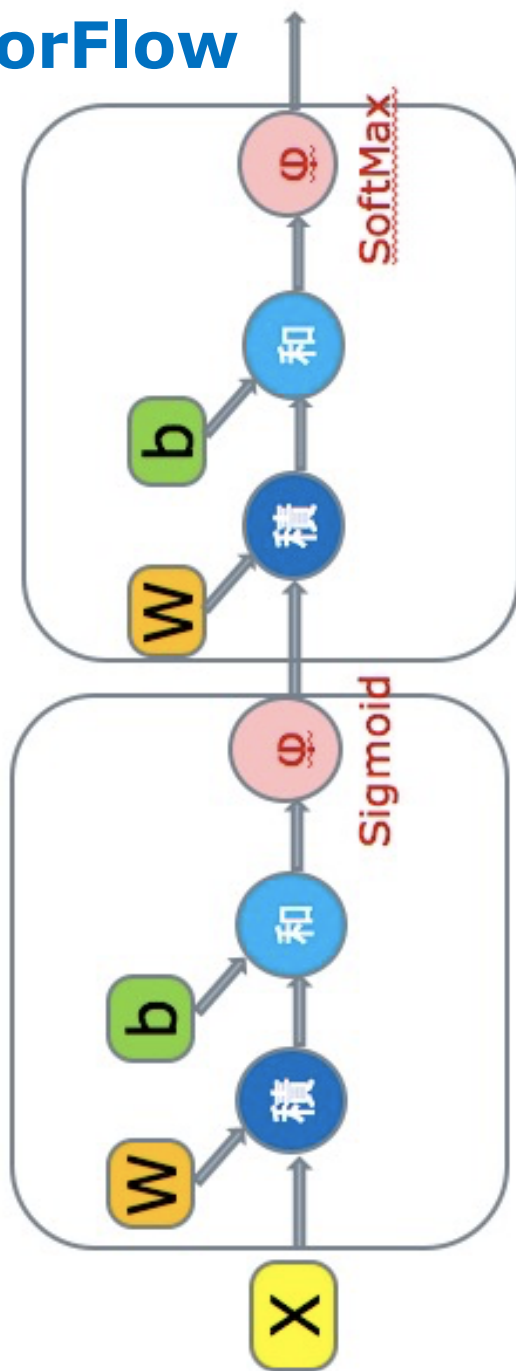
入力層

隠れ層

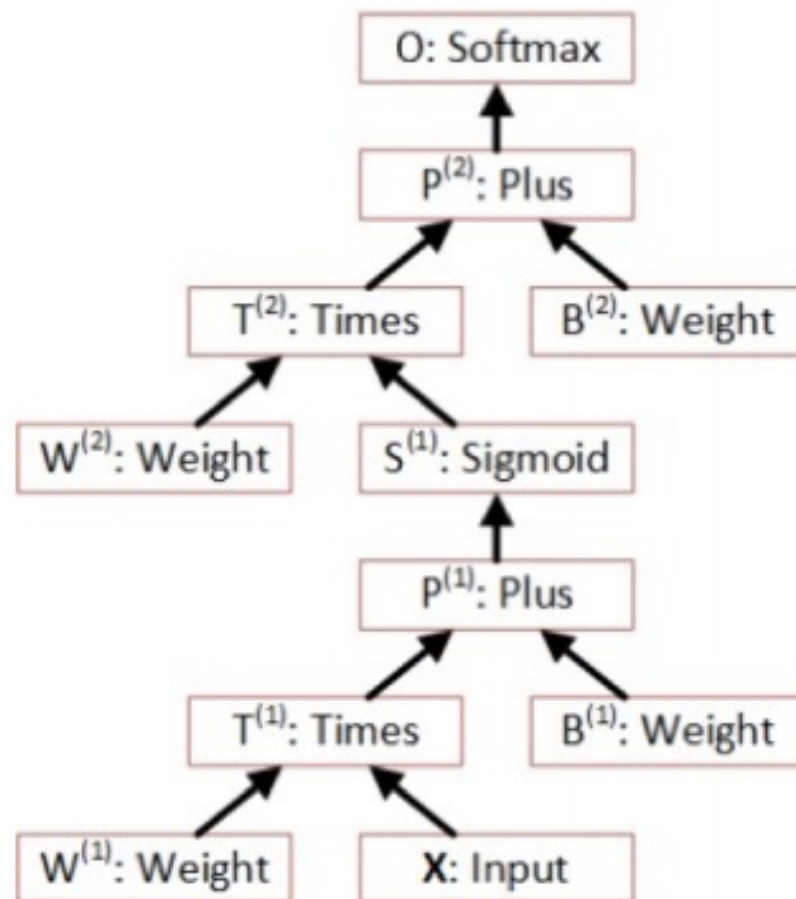
出力層

こっちのグラフの方が、ずっとわかりやすい！

TensorFlow

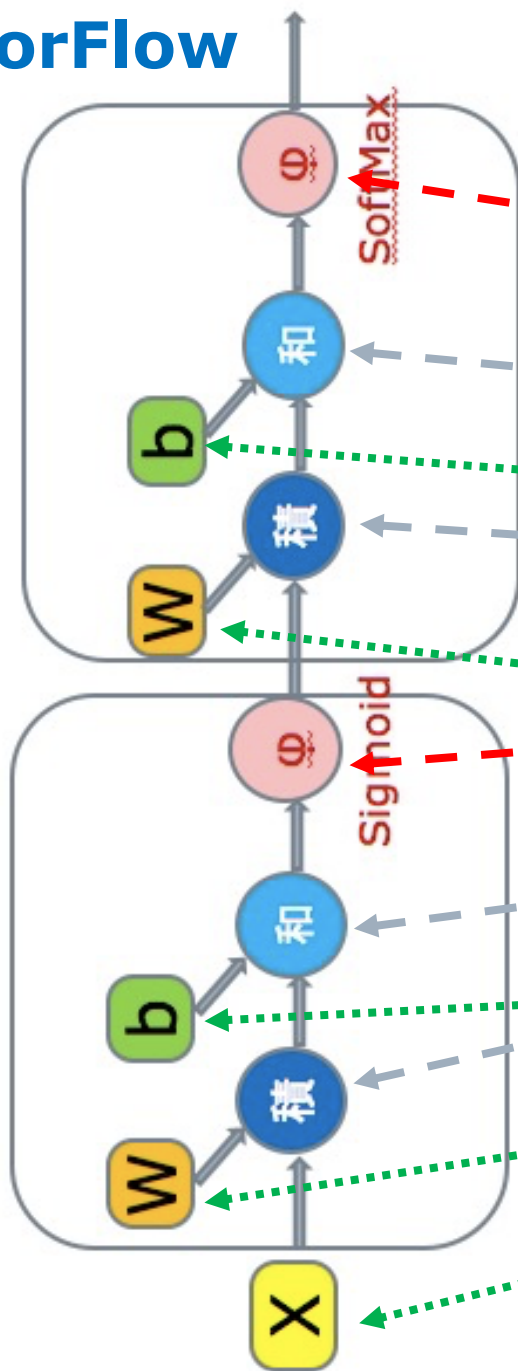


CNTK

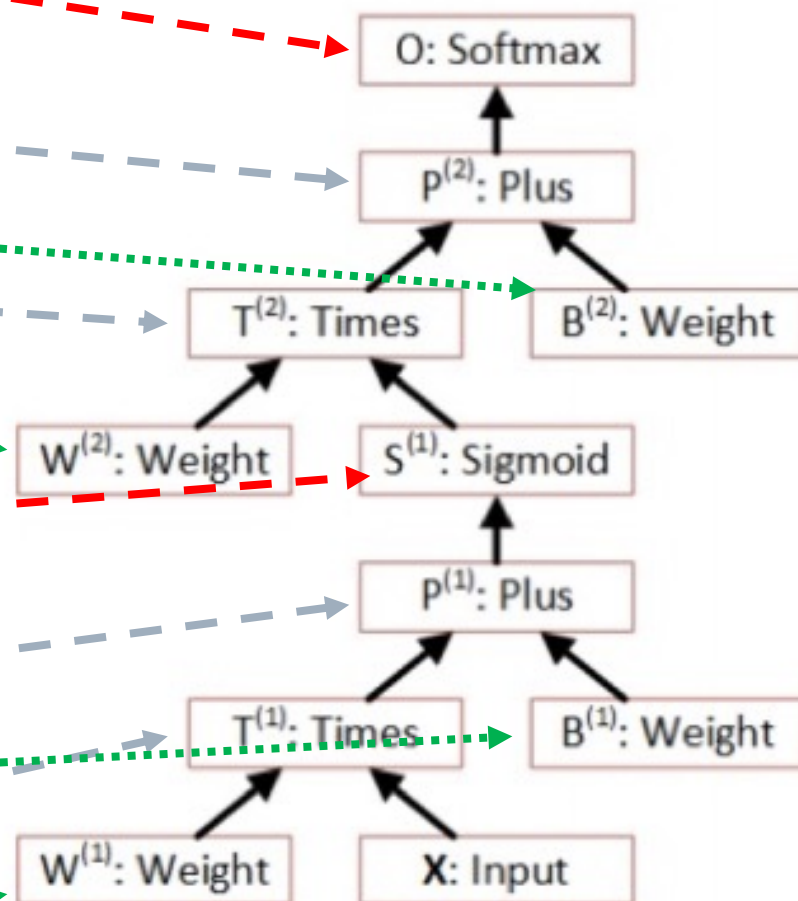


これらのグラフは、同じものである

TensorFlow

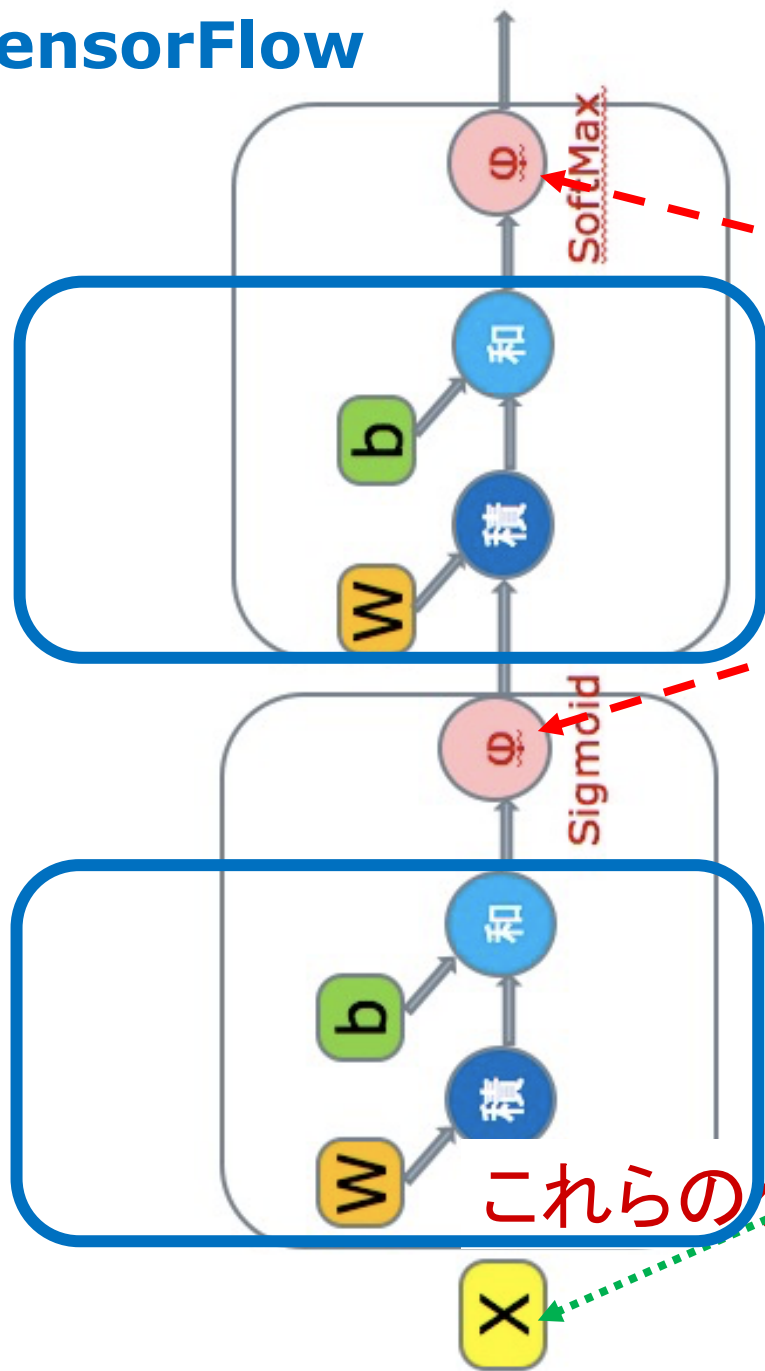


CNTK

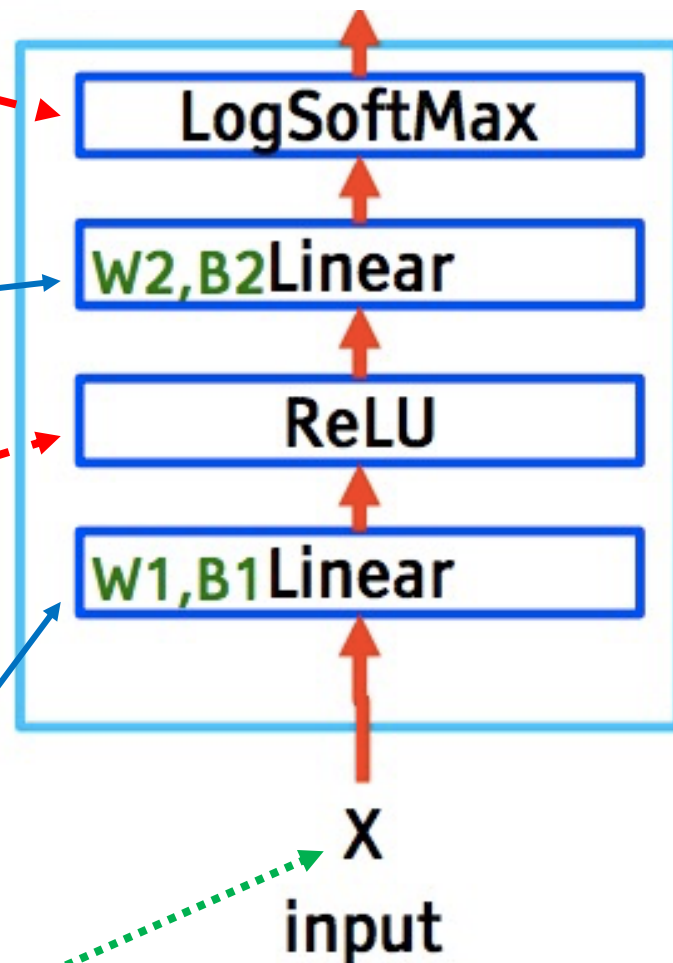


これらのグラフは、同じものである

TensorFlow



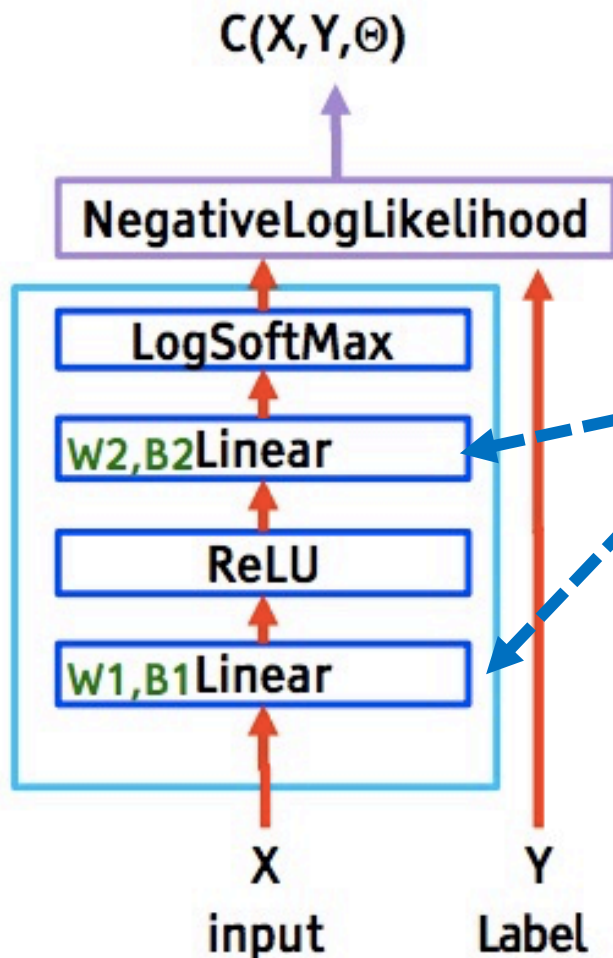
Torch7



これらのグラフは、ほぼ、同じものである

Torch7でのDNNモジュールの記述例

- All major deep learning frameworks use modules (inspired by SN/Lush, 1991)
 - Torch7, Theano, TensorFlow....



```
-- sizes
ninput = 28*28 -- e.g. for MNIST
nhidden1 = 1000
noutput = 10
```

```
-- network module
net = nn.Sequential()
net:add(nn.Linear(ninput, nhidden))
net:add(nn.Threshold())
net:add(nn.Linear(nhidden, noutput))
net:add(nn.LogSoftMax())
```

← DNN

```
-- cost module
cost = nn.ClassNLLCriterion()
```

```
-- get a training sample
input = trainingset.data[k]
target = trainingset.labels[k]
```

```
-- run through the model
output = net:forward(input)
c = cost:forward(output, target)
```

TensorFlowでのDNNモジュールの記述例

```
# Hidden 1
with tf.name_scope('hidden1'):
    weights = tf.Variable(
        tf.truncated_normal([IMAGE_PIXELS, hidden1_units],
                             stddev=1.0 / math.sqrt(float(IMAGE_PIXELS))),
        name='weights')
    biases = tf.Variable(tf.zeros([hidden1_units]),
                          name='biases')
    hidden1 = tf.nn.relu(tf.matmul(images, weights) + biases)
# Hidden 2
with tf.name_scope('hidden2'):
    weights = tf.Variable(
        tf.truncated_normal([hidden1_units, hidden2_units],
                             stddev=1.0 / math.sqrt(float(hidden1_units))),
        name='weights')
    biases = tf.Variable( tf.zeros([hidden2_units]),
                          name='biases')
    hidden2 = tf.nn.relu(tf.matmul(hidden1, weights) + biases)
```

```
# Linear
```

```
....
....
```

DNNモジュールが、 $\phi(X \cdot W + b)$ の形をしていることは、簡単に確認できる。



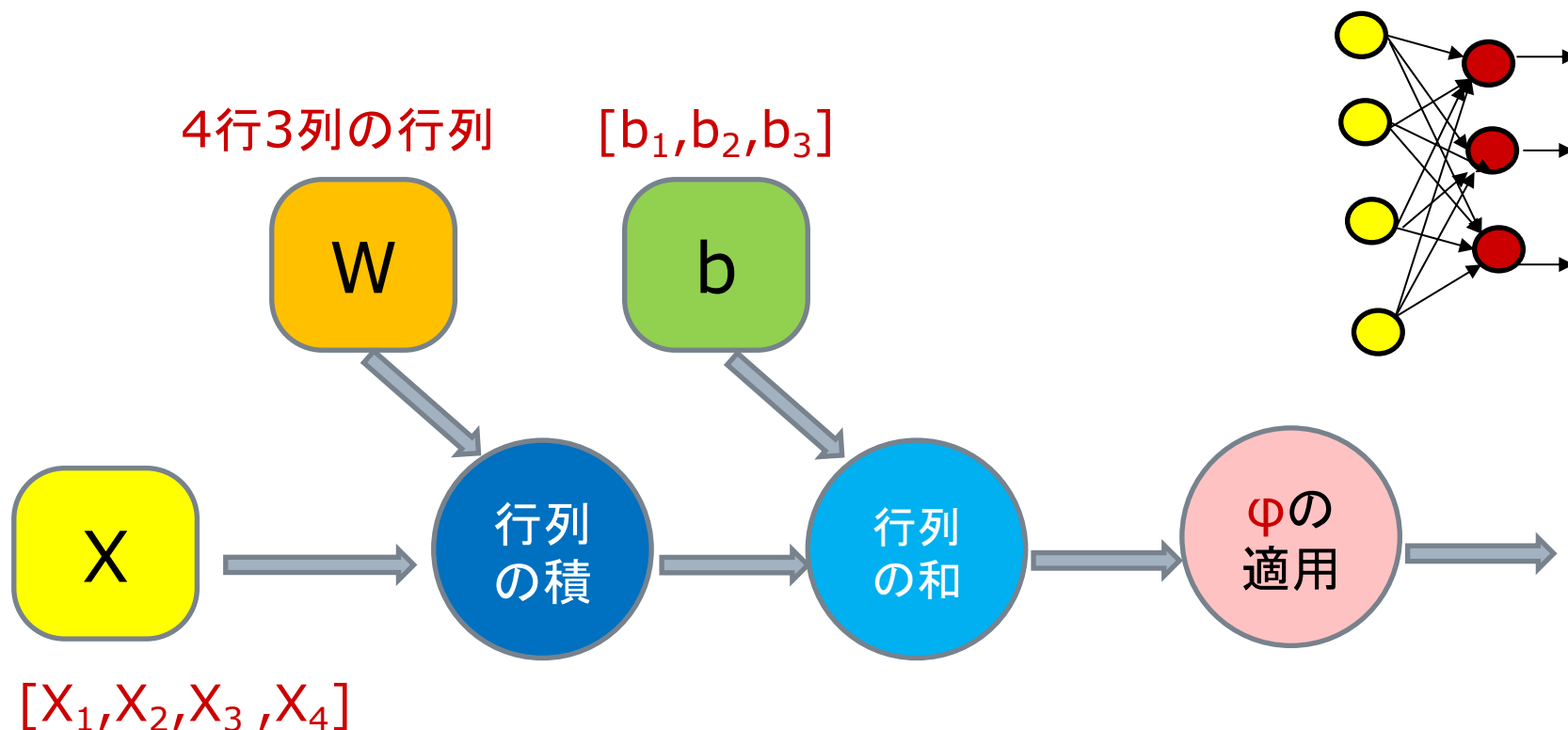
グラフを流れる量 -- テンソル

グラフを流れる量 -- テンソル

旧来のグラフの沢山の線は、どこへ行ったのか？

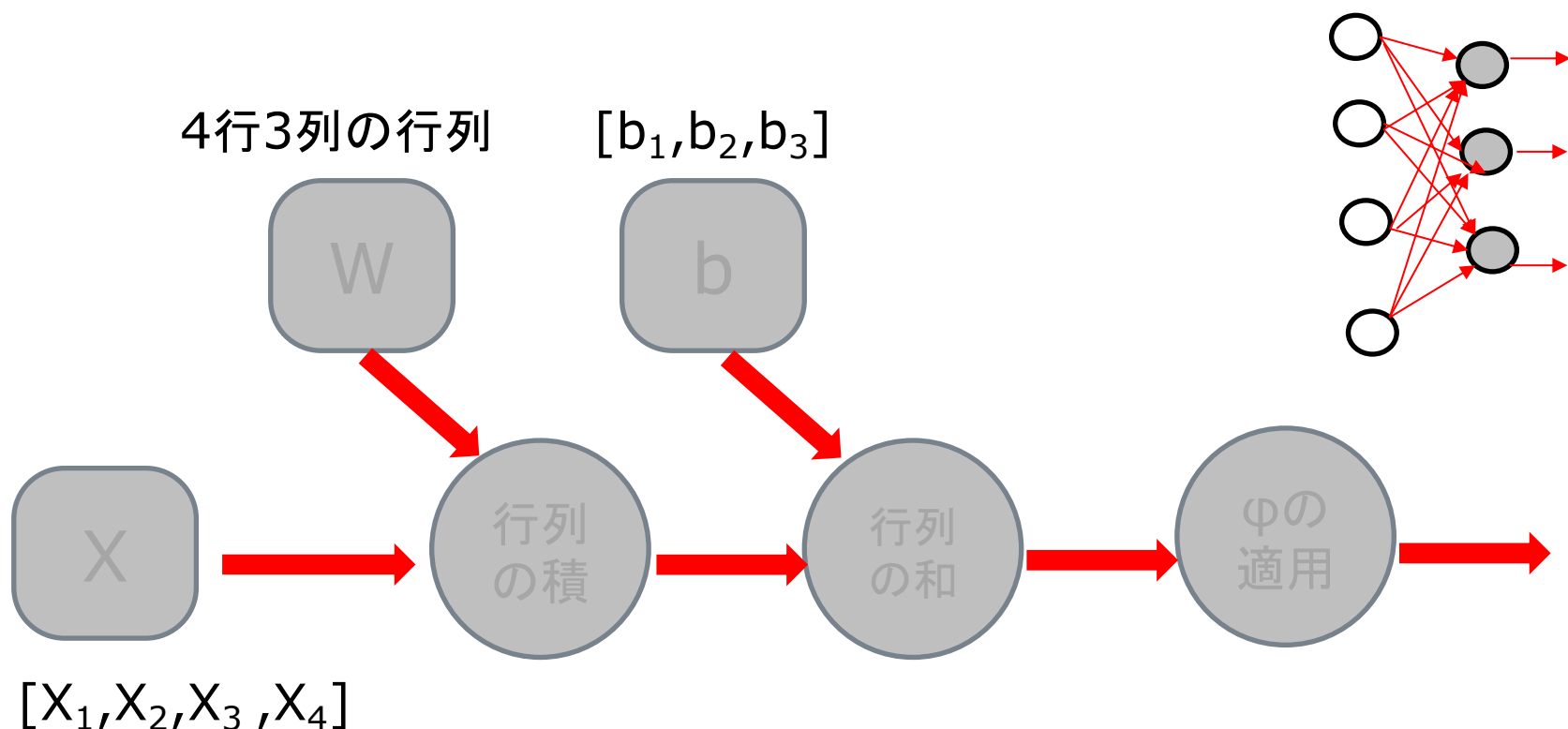
TensorFlowのグラフでは、グラフを流れるデータは、すべて「テンソル」にまとめられている。

4つの入力を受け取り、3つの出力を返す、3つのニューロン
からなる「層」を考えてみよう。



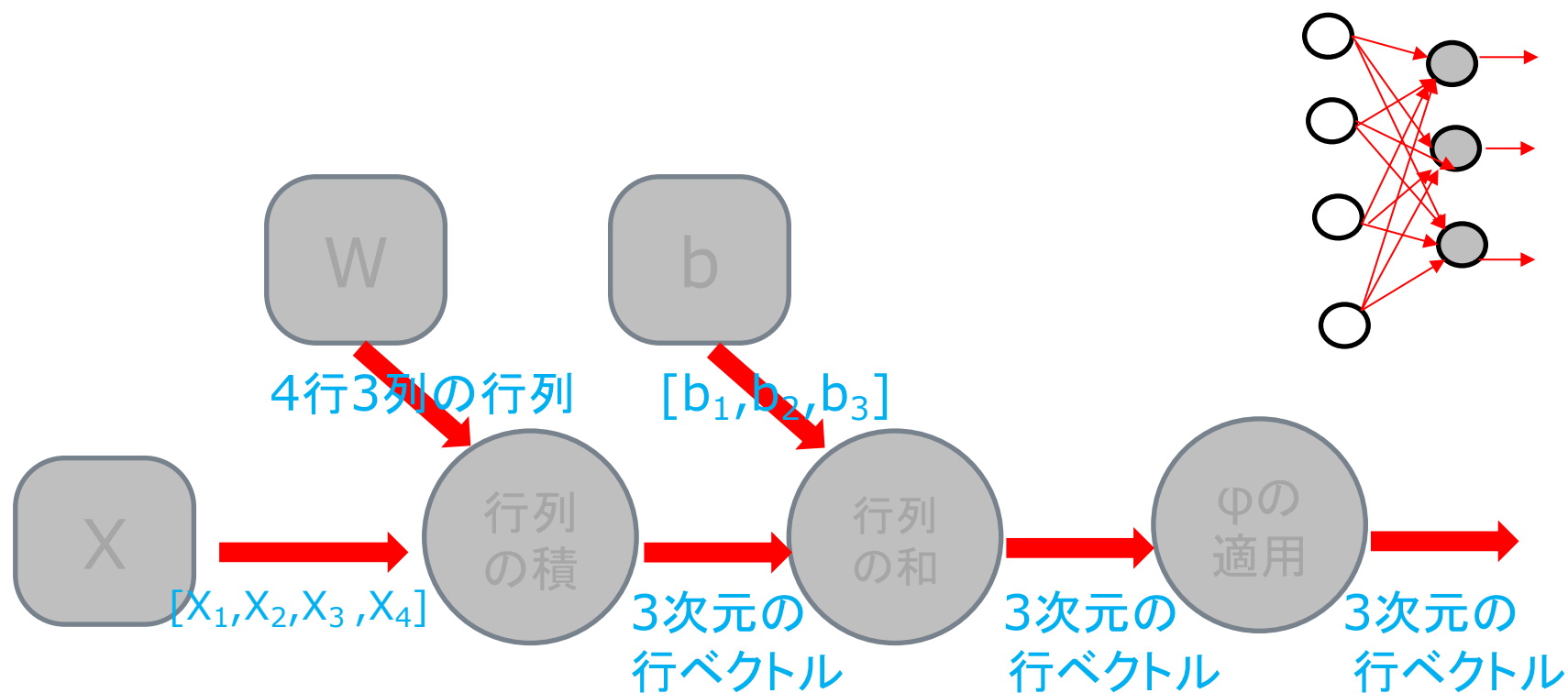
ただ、ここでは、ノードではなく辺の方に注目してみよう。
どのようなタイプのデータが、辺の上を流れるかを考えよう。

4つの入力を受け取り、3つの出力を返す、3つのニューロンの「層」



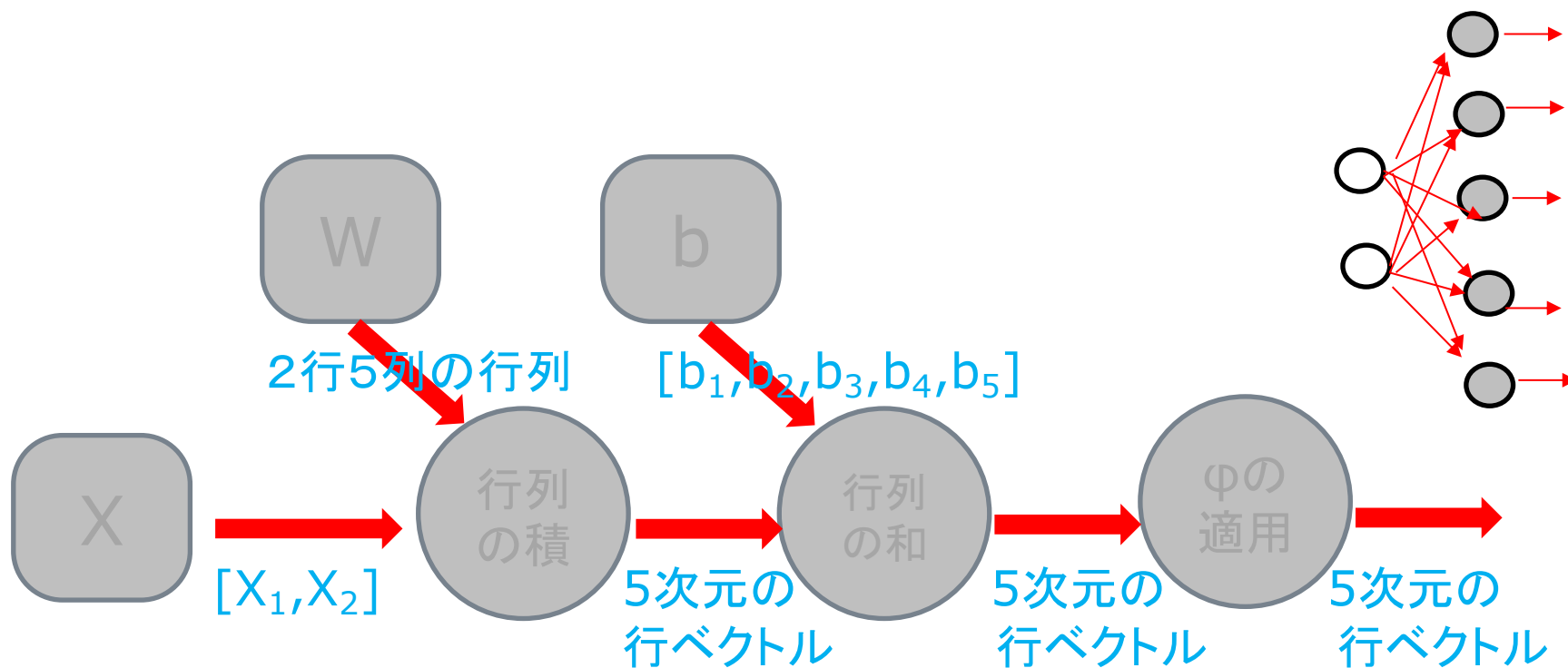
TensorFlowでは、こうしたデータをすべて**テンソル**と呼んでいる。
どのような、タイプのデータか見てみよう。

4つの入力を受け取り、3つの出力を返す、3つのニューロンの「層」



テンソルには、いろいろなタイプがある。

2つの入力を受け取り、5つの出力を返す、5つのニューロンの「層」



テンソルには、いろいろなタイプがある。

TensorFlowとテンソル

- TensorFlowのプログラムは、すべてのデータを表現するものとしてテンソル・データ構造を利用する。
 - TensorFlowのテンソルは、n次元の配列あるいはリストだと考えることができる。
 - テンソルは、静的な型と動的な次元を持っている。
 - 計算グラフのノード間を移動するのは、テンソルだけである。
-

テンソルのランク(階数)

- TensorFlowのシステムでは、テンソルは「ランク」と呼ばれる次元の単位で記述される。
- テンソルのランクは、行列のランクとは異なるものである。
- テンソルのランクは、(時には、位数(order)とか度数(degree)とかn次元とも呼ばれることがあるのだが)、テンソルの次元の数である。
- 例えば、Pythonのリストで定義された次のテンソルのランクは 2である。

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

テンソルのランク(階数)

- ランク2のテンソルで、われわれが典型的に考えるのは「行列」である。
 - ランク1のテンソルは、「ベクトル」である。
 - ランク2のテンソルに対しては、 $t[i,j]$ という構文で、任意の要素にアクセスできる。
 - ランク3のテンソルには、 $t[i,j,k]$ でアクセスする必要がある。
-

テンソルのランク(階数)

ランク	数学的実体	Pythonでの表記
0	スカラー(大きさのみ)	<code>s = 483</code>
1	ベクトル(大きさと向き)	<code>v = [1.1, 2.2, 3.3]</code>
2	行列 (数字のテーブル)	<code>m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]</code>
3	3-テンソル	<code>t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]</code>
n	n-Tensor	<code>....</code>

テンソルの形

- TensorFlowのドキュメントでは、テンソルの次元を記述するのに、ランク、形、次元数の三つを用いる。

Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn]	n-D	A tensor with shape [D0, D1, ... Dn].

Part II

Convolutional Neural Network

CNN以前の ニューラル・ネットワーク

まず最初に、CNN以前のニューラル・ネットワークを振り返ってみよう。

ニューロンの働きのモデルとしての ニューラル・ネットワーク

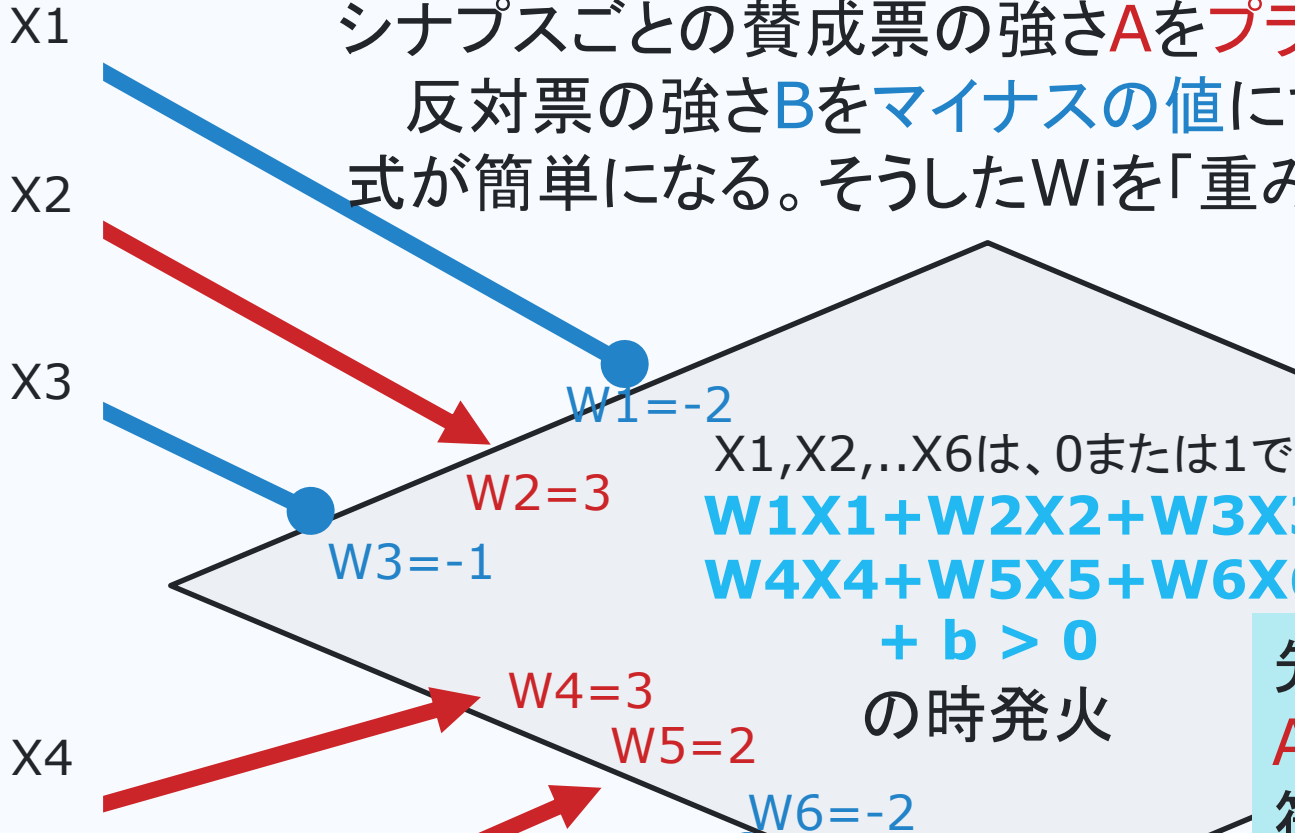
以前のマルレク「ニューラル・ネットワーク入門講座」でも見たように、ニューラル・ネットワークは、ニューロンの働きのモデルとして構想された。

ここでは、ニューロンと、信号の関係に注目しよう。

復習

重み: W_i

シナプスごとの賛成票の強さAをプラスの値に
反対票の強さBをマイナスの値にすると、
式が簡単になる。そうした W_i を「重み」という。

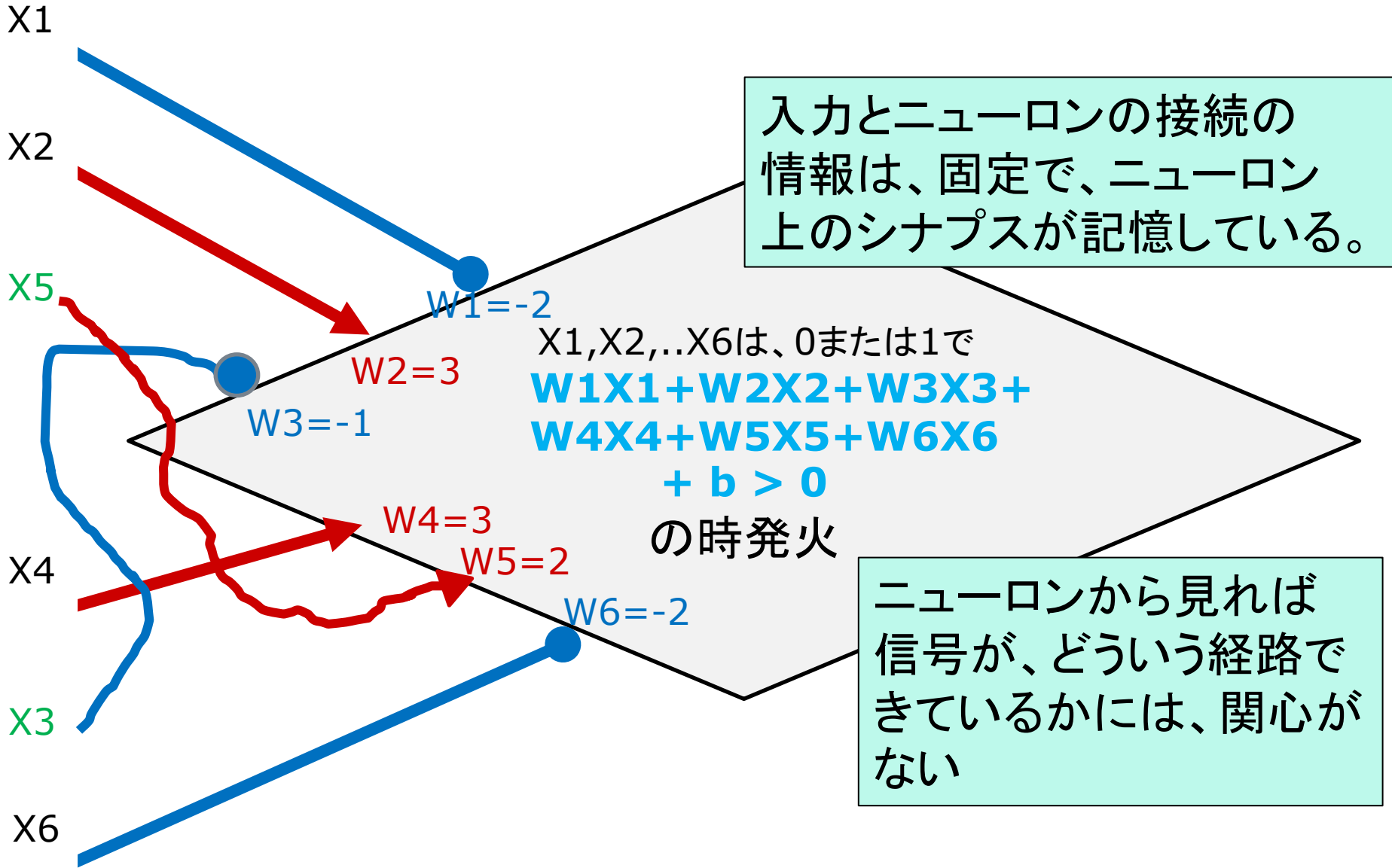


バイアス: b

ついでに、閾値 C の符号を逆にすれば、
式はもっと簡単になる。それを「バイアス」という。

先の次式を想起せよ
 $A - B - C > 0$
符号を変えると
 $A + B' + C' > 0$
の形になる。

入力ベクトルの並びの意味



一個のニューロンの場合

重み $W=[W_1, W_2, W_3, W_4, W_5, W_6]$

バイアス b

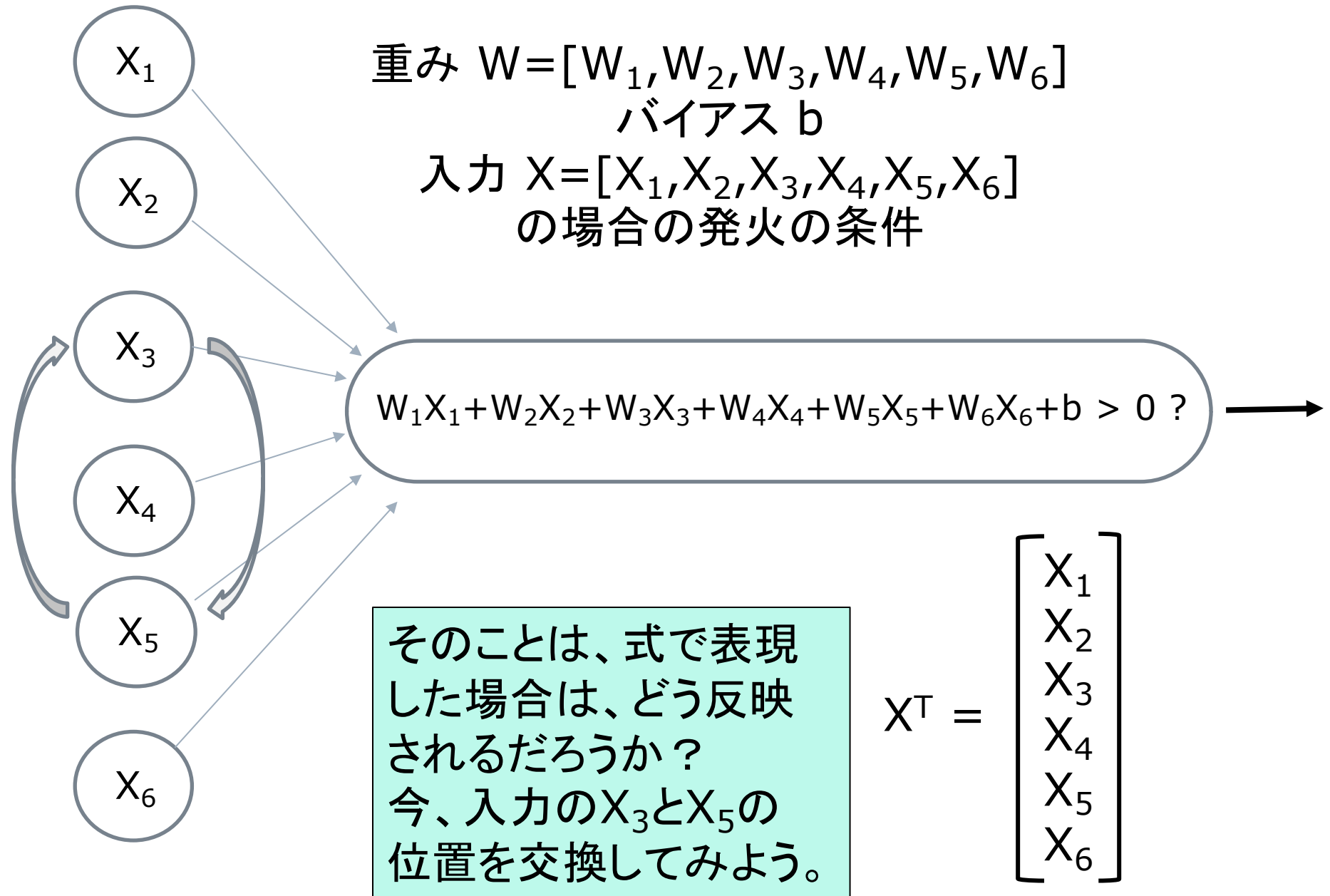
入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$

の場合の発火の条件

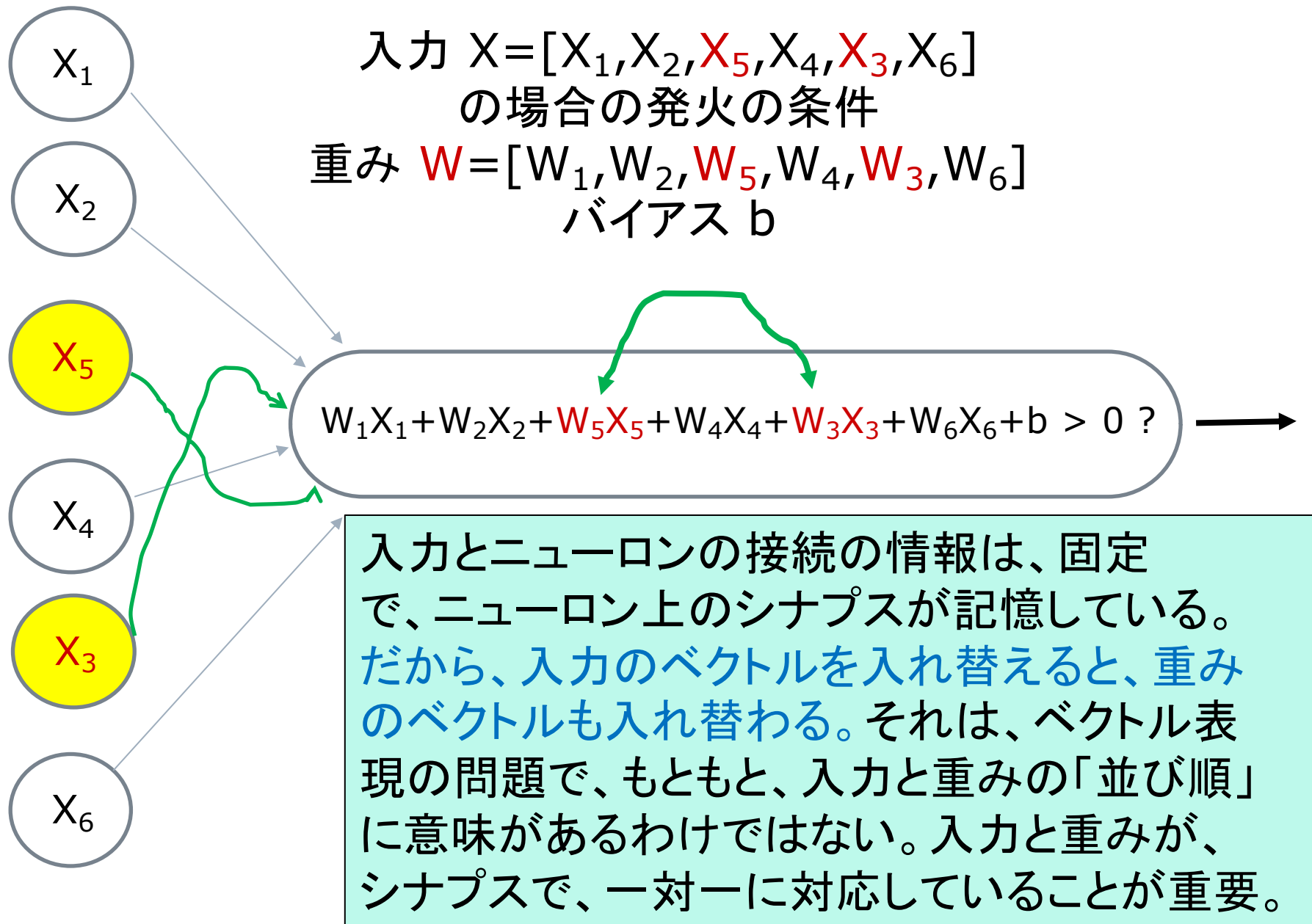
$$W_1X_1+W_2X_2+W_3X_3+W_4X_4+W_5X_5+W_6X_6+b > 0 ?$$

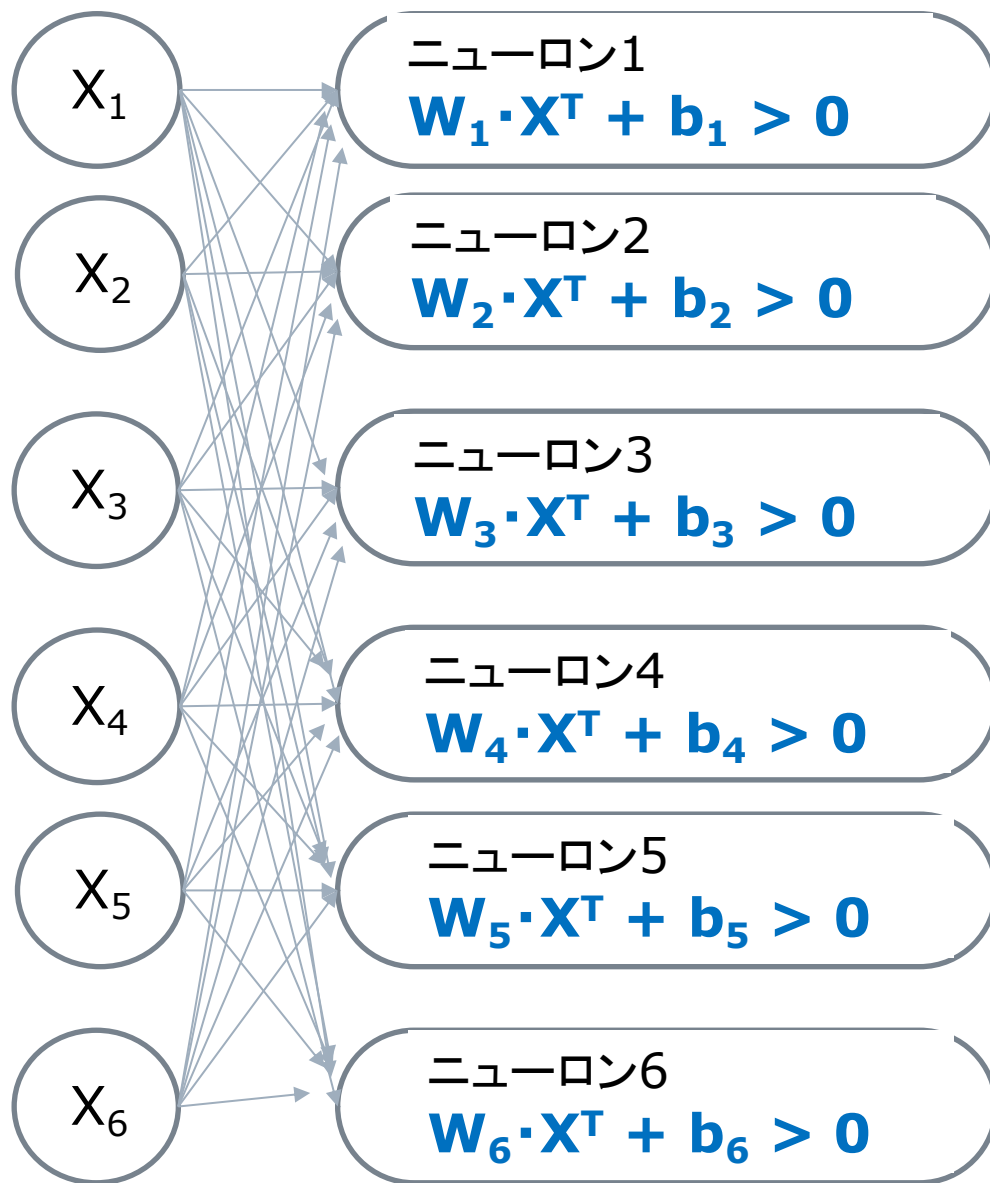
そのことは、式で表現
した場合は、どう反映
されるだろうか？
今、入力の X_3 と X_5 の
位置を交換してみよう。

$$X^T = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{bmatrix}$$



一個のニューロンの場合



入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$ 

$$\text{値: } \cancel{W_{11}X_1} + \cancel{W_{12}X_2} + \cancel{W_{13}X_3} + W_{14}X_4 + W_{15}X_5 + W_{16}X_6 + b_1$$

$$\text{値: } W_{21}X_1 + W_{22}X_2 + W_{23}X_3 + W_{24}X_4 + W_{25}X_5 + W_{26}X_6 + b_2$$

$$\text{値: } W_{31}X_1 + W_{32}X_2 + W_{33}X_3 + W_{34}X_4 + W_{35}X_5 + W_{36}X_6 + b_3$$

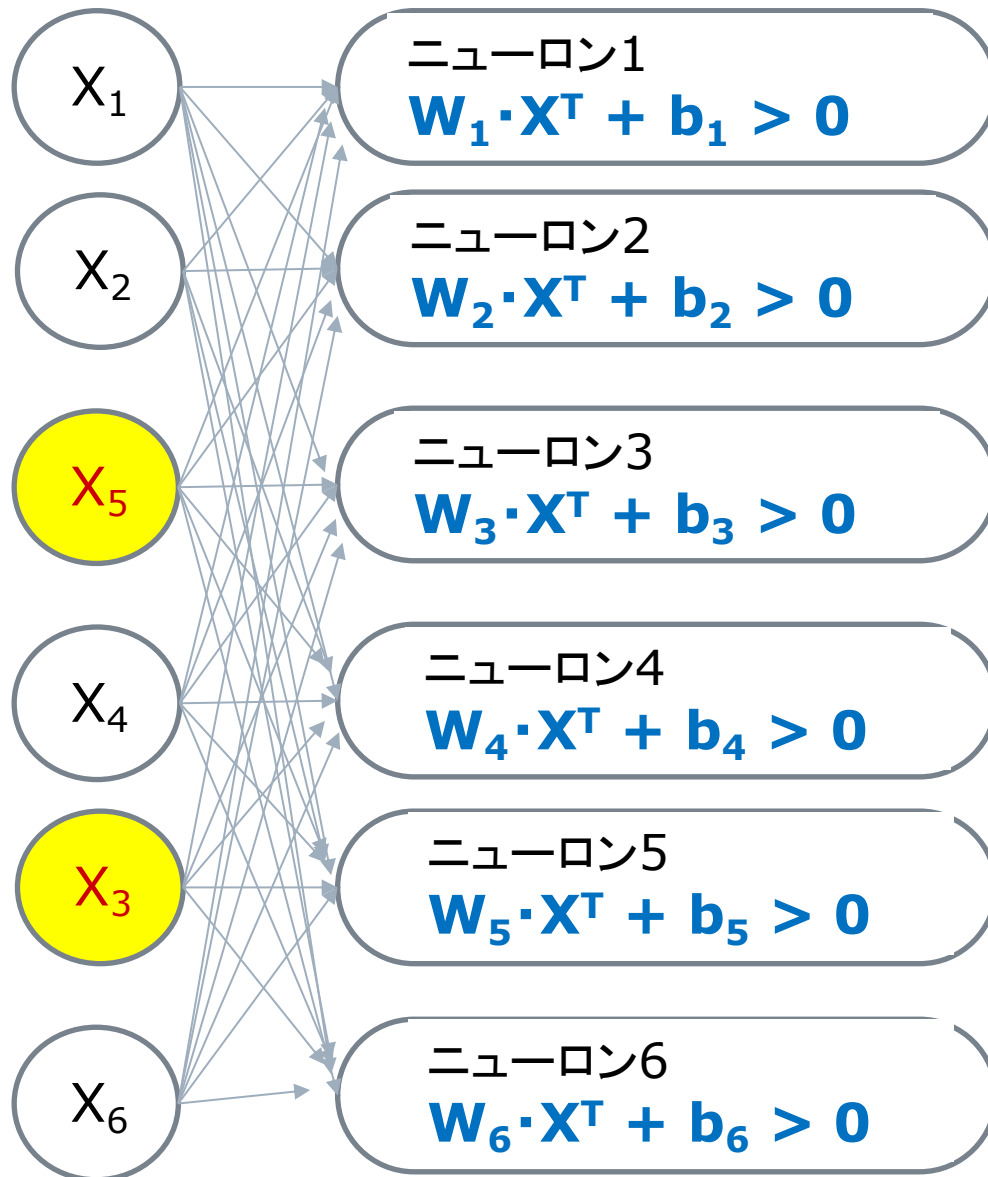
$$\text{値: } W_{41}X_1 + W_{42}X_2 + W_{43}X_3 + W_{44}X_4 + W_{45}X_5 + W_{46}X_6 + b_4$$

$$\text{値: } \cancel{W_{51}X_1} + \cancel{W_{52}X_2} + \cancel{W_{53}X_3} + W_{54}X_4 + W_{55}X_5 + W_{56}X_6 + b_5$$

$$\text{値: } \cancel{W_{61}X_1} + \cancel{W_{62}X_2} + \cancel{W_{63}X_3} + W_{64}X_4 + W_{65}X_5 + W_{66}X_6 + b_6$$

層としてのニューロンの場合

入力 $X=[X_1, X_2, X_5, X_4, X_3, X_6]$



$$\text{値} = W_{11}X_1 + W_{12}X_2 + W_{13}X_3 + W_{14}X_4 + W_{15}X_5 + W_{16}X_6 + b_1$$

$$\text{値} = W_{21}X_1 + W_{22}X_2 + W_{23}X_3 + W_{24}X_4 + W_{25}X_5 + W_{26}X_6 + b_2$$

$$\text{値} = W_{31}X_1 + W_{32}X_2 + W_{33}X_3 + W_{34}X_4 + W_{35}X_5 + W_{36}X_6 + b_3$$

$$\text{値} = W_{41}X_1 + W_{42}X_2 + W_{43}X_3 + W_{44}X_4 + W_{45}X_5 + W_{46}X_6 + b_4$$

$$\text{値} = W_{51}X_1 + W_{52}X_2 + W_{53}X_3 + W_{54}X_4 + W_{55}X_5 + W_{56}X_6 + b_5$$

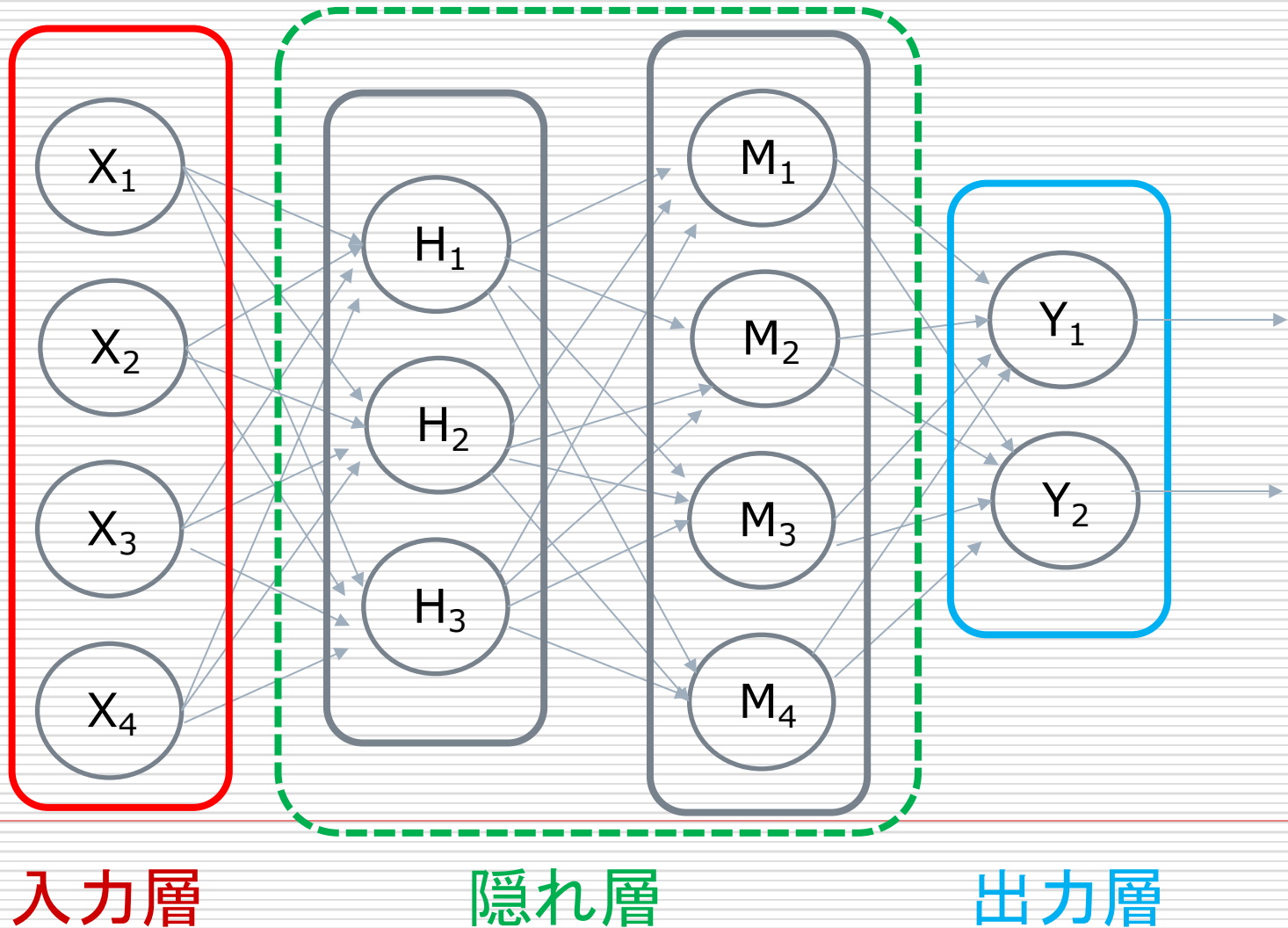
$$\text{値} = W_{61}X_1 + W_{62}X_2 + W_{63}X_3 + W_{64}X_4 + W_{65}X_5 + W_{66}X_6 + b_6$$

層としても、入力ベクトルの入れ替えは、重みの入れ替えを必要とし、影響を与えない。

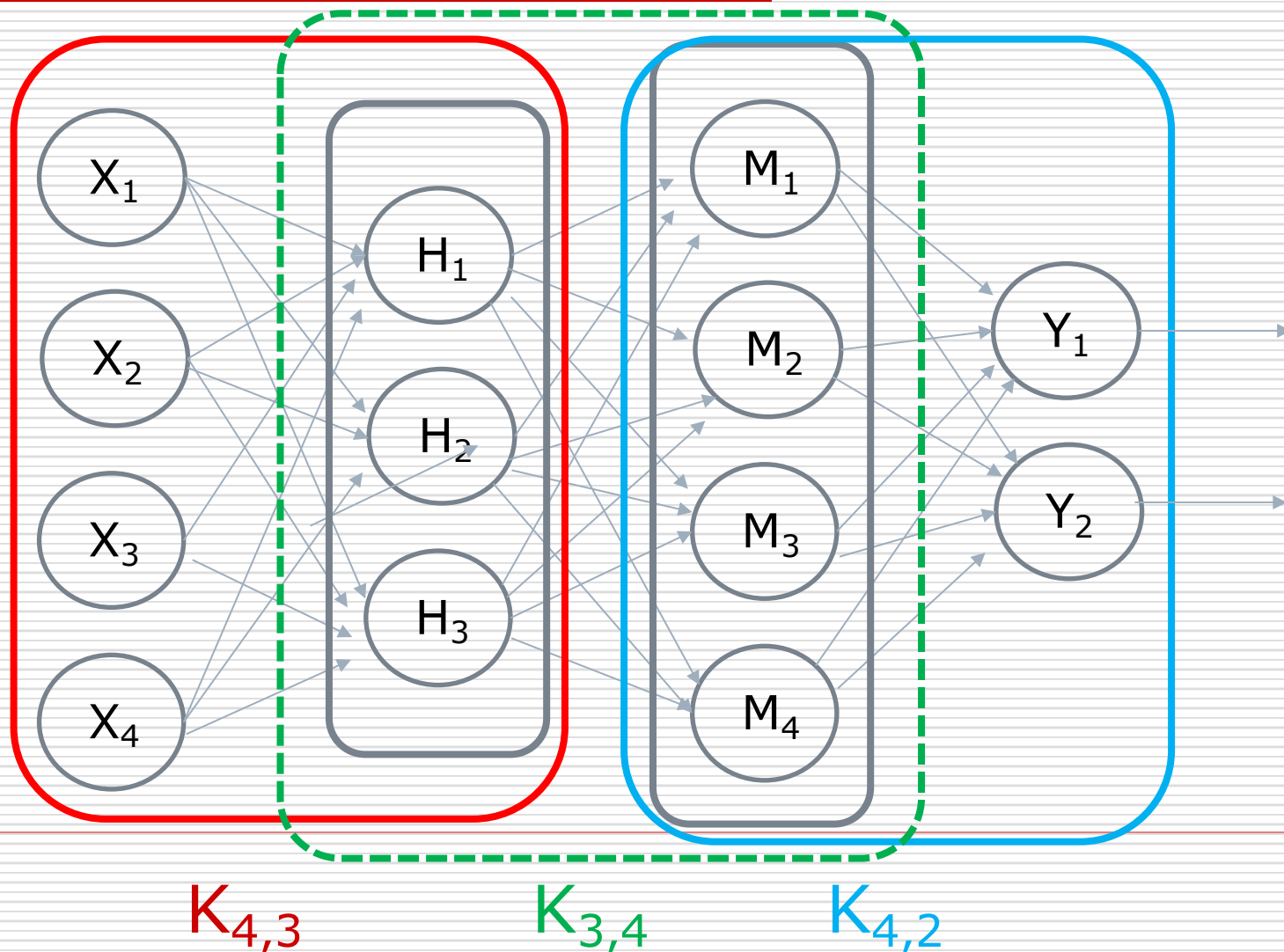
Full Connectの性質を考える

ここでは、もう少し詳しく、Full Connectなネットワークの性質を考えてみよう。Full Connectのようなノード間の密な結合は、ネットワークを記述するパラメータの、増大をもたらすことになる。

Full Connectな ニューラル・ネットワーク

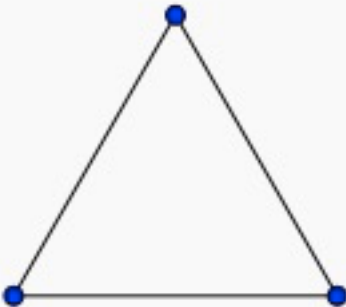
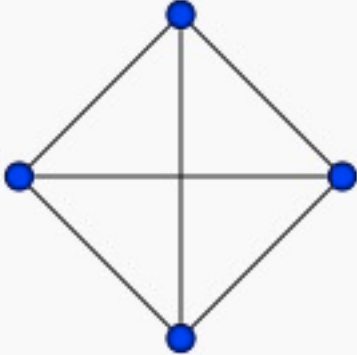


Full Connectなネットワークでは、隣り合う層は「完全二部グラフ」で表現される

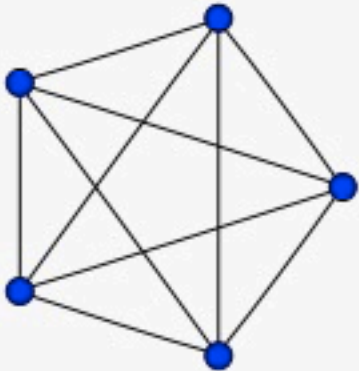
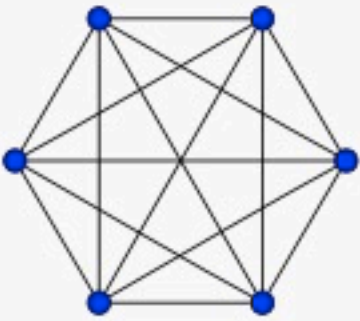
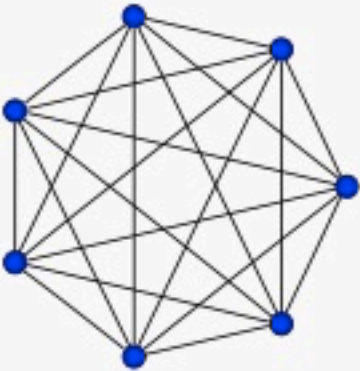
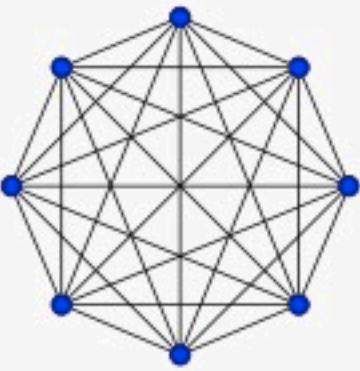
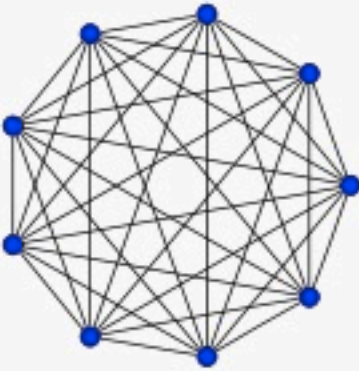
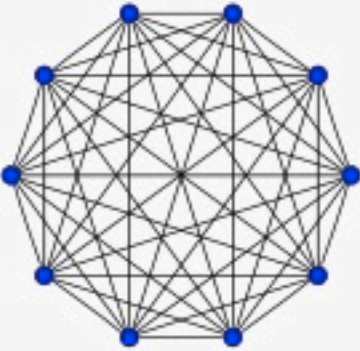
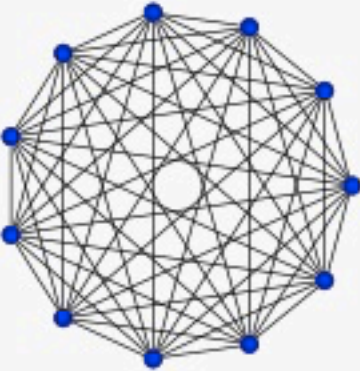
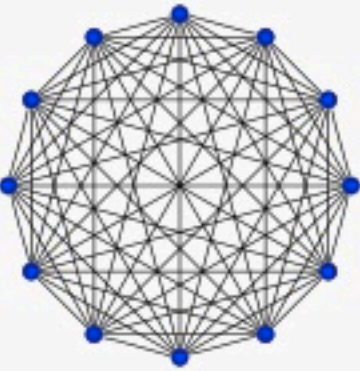


完全グラフ

- 完全グラフ (complete graph) とは任意の2頂点間に辺があるグラフのことを指す。

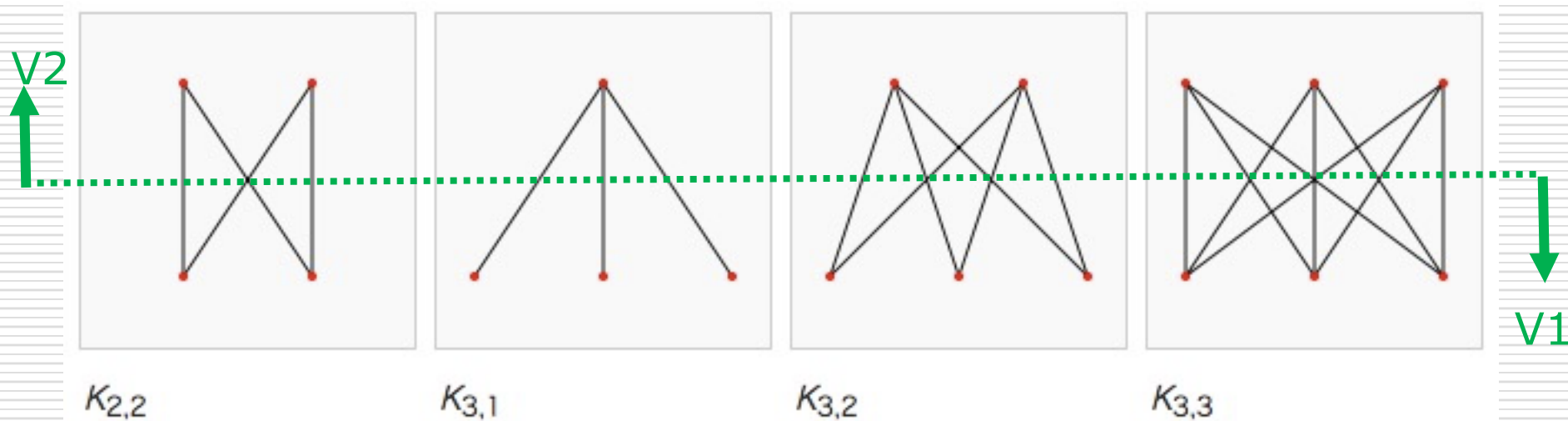
$K_1: 0$	$K_2: 1$	$K_3: 3$	$K_4: 6$
			

完全グラフ

$K_5: 10$	$K_6: 15$	$K_7: 21$	$K_8: 28$
			
$K_9: 36$	$K_{10}: 45$	$K_{11}: 55$	$K_{12}: 66$
			

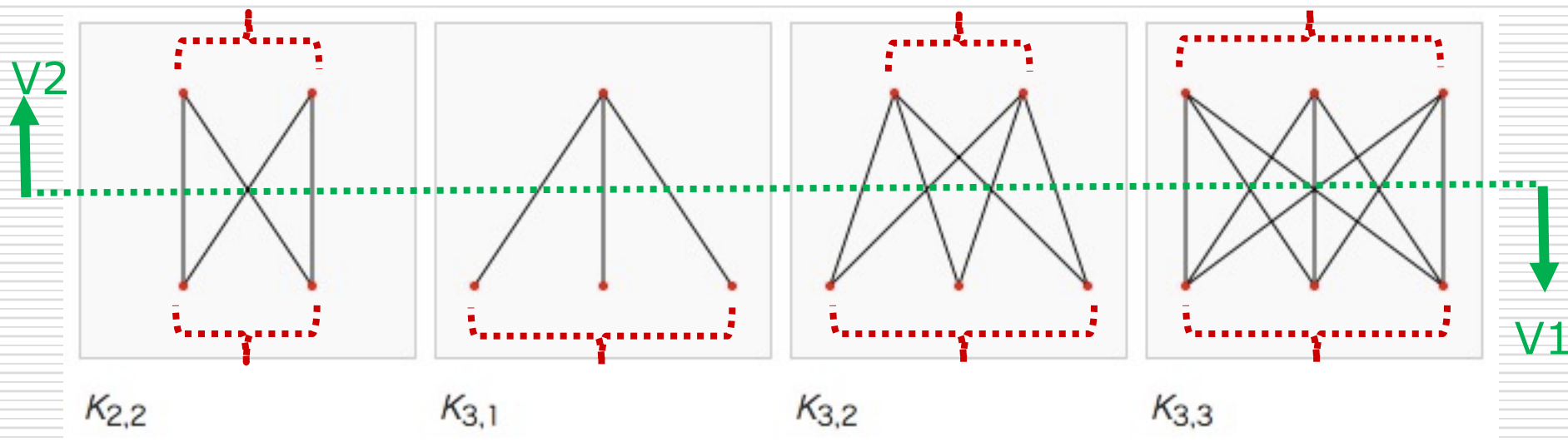
完全二部グラフ

- 頂点が二つの集合 $V1$, $V2$ に分かれていて、 $V1$ に属する任意の頂点と、 $V2$ に属する任意の頂点との間に辺があるもの。



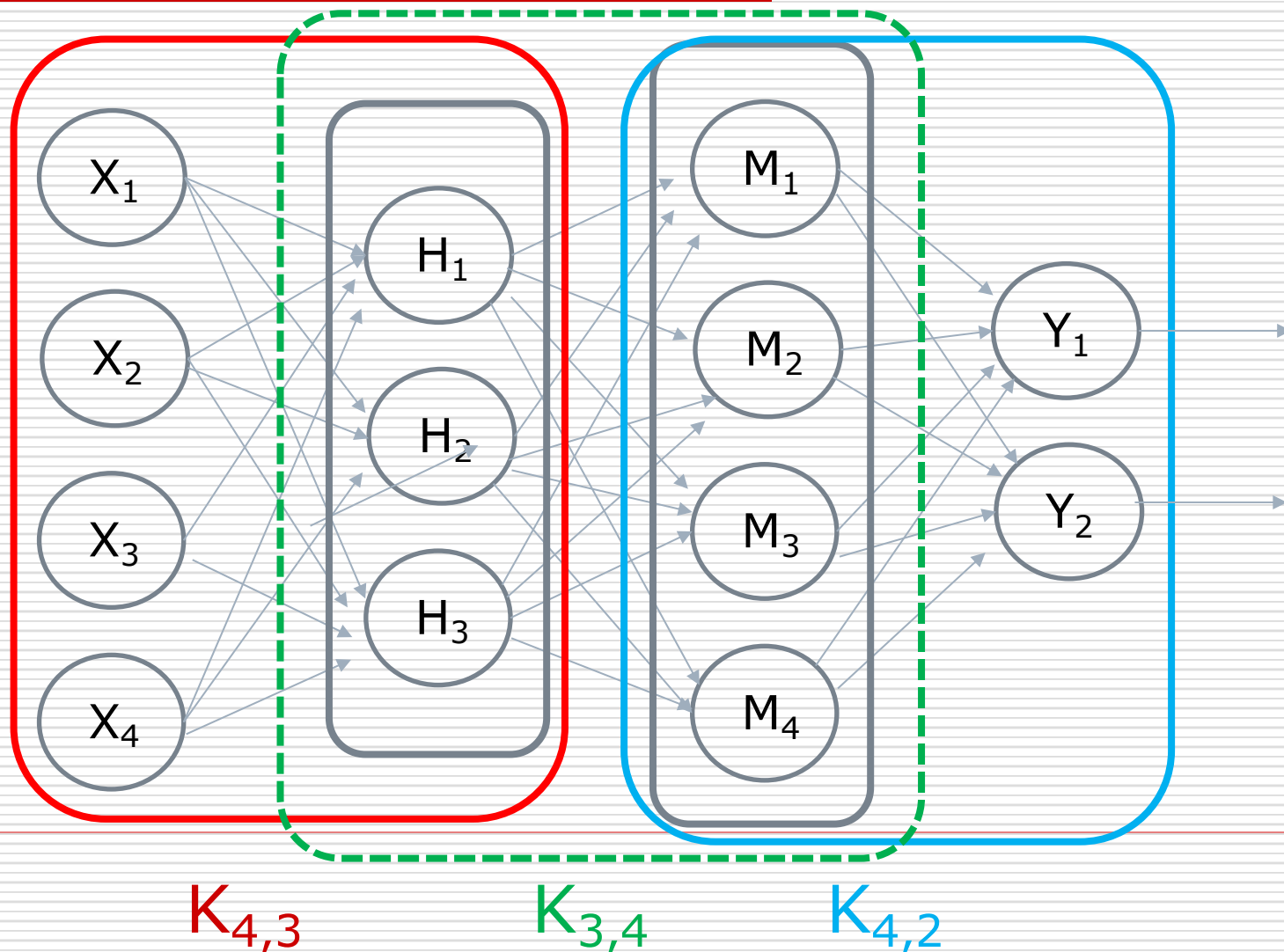
完全二部グラフ

- 完全二部グラフには、もう一つ、重要な特徴がある。それは、各集合内の頂点同士の間には辺が無いということである。

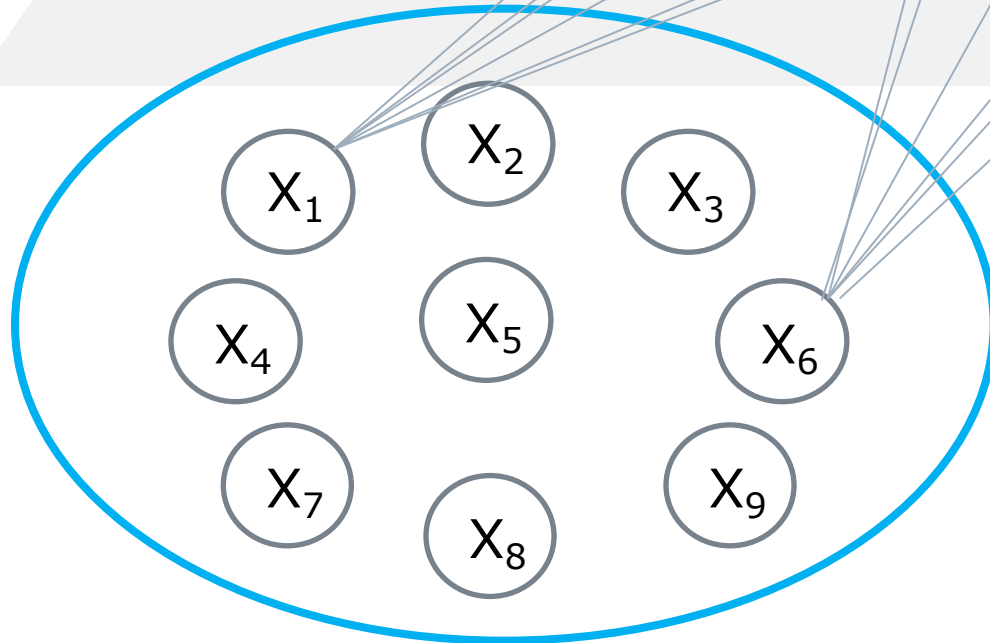
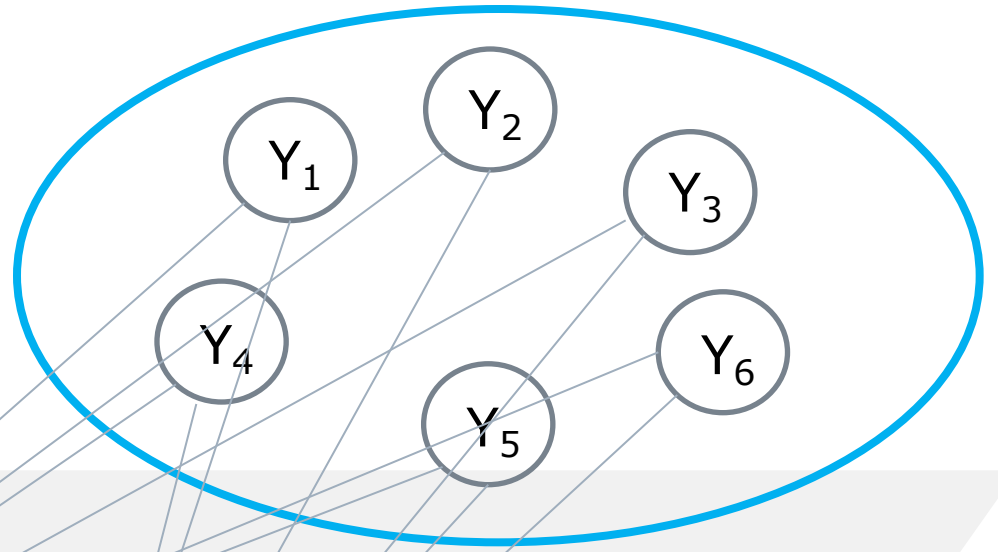


- Full Connectなニューラル・ネットワークの、隣り合う層のあいだのグラフは、完全二部グラフである。

Full Connectなネットワークでは、隣り合う層は「完全二部グラフ」で表現される



Full ConnectするX層とY層の、要素の接続の様子。
ここでは、便宜的に、X層の
ノード X_1 と X_6 の接続のみを
図示した。



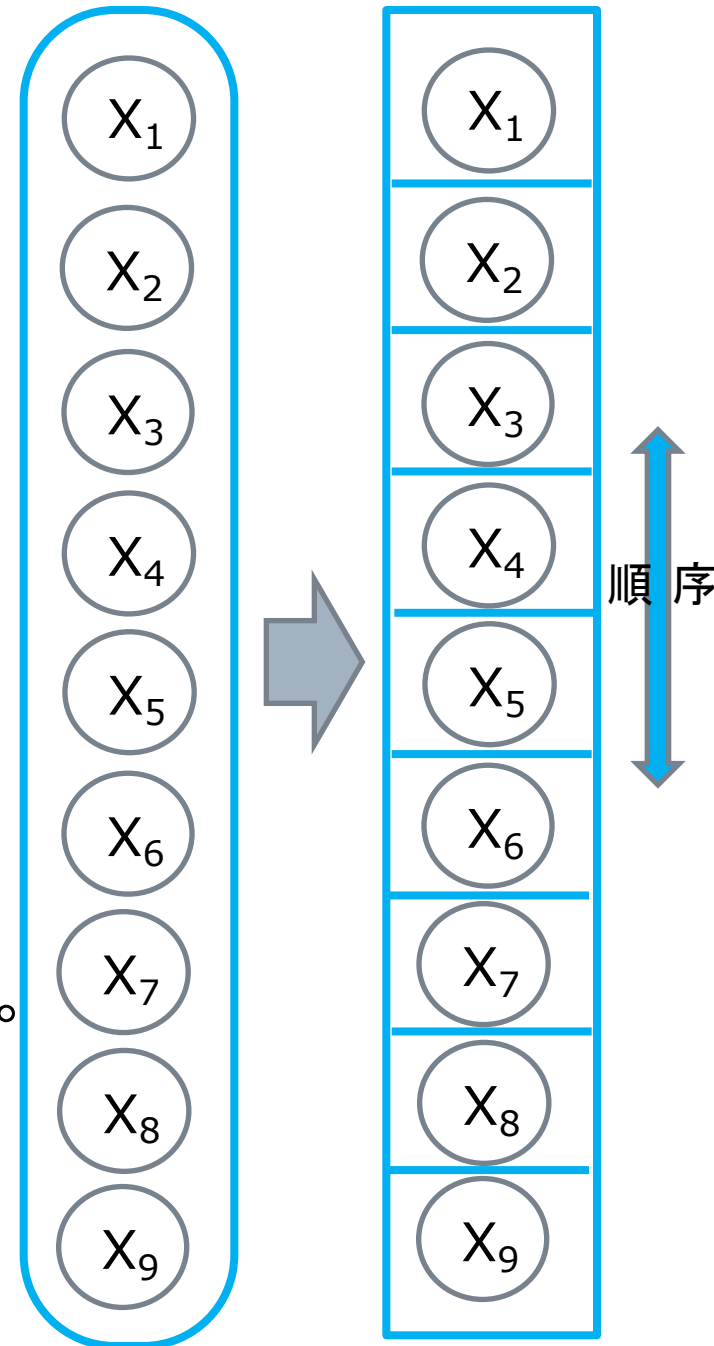
パラメーターの数は、
それぞれの層のノードの
数をかけたものなので、
ネットワークが複雑なもの
になるつれて、急速に大き
くなりうる。

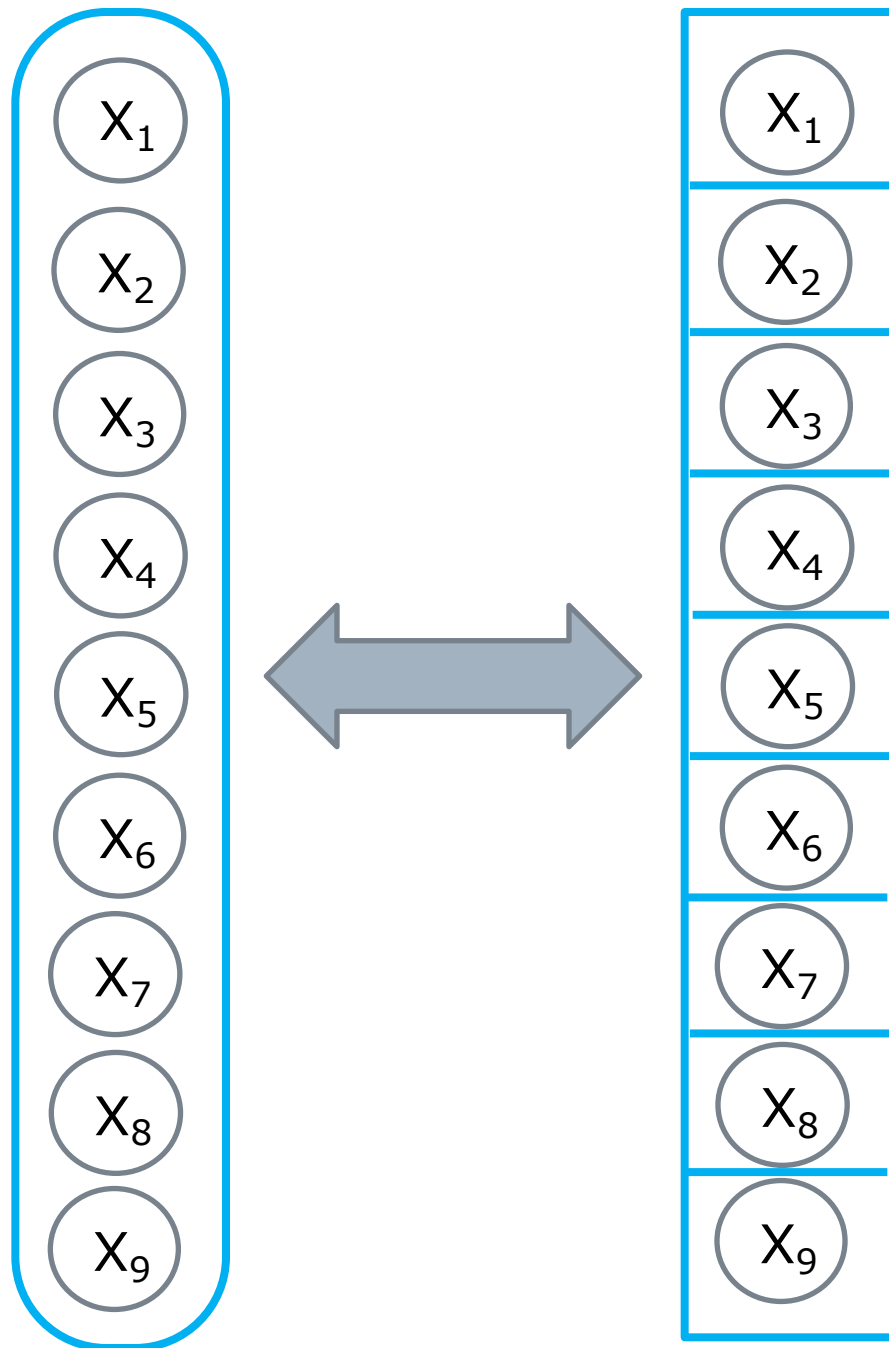
Full Connectのネットワークの 入力層は、順序を持つ「配列」か？

Full Connectなネットワークでの入力層は、順序を持つ「配列」なのだろうか？ ここでは、そうではないことを述べる。それは、そうしたネットワークでは、入力層は、構造を持たないので「画像」データのもつ構造的な情報を、直接には、正確に反映していないことを意味する。

層の中のノードのイメージ

- Full Connectなネットワークの層の中のノードの集まりを、我々は、次のように、ノードが順序を持つ一次元のベクトルと考えることが多い。
- $X=[X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9]$ といった、入力のベクトル表記や、層全体の重み行列 W_{ij} を考えて、層全体の出力を $\varphi(X \cdot W + b)$ で計算するなど、そうした形式化は有用である。
- ただ、それは、対象の一面である。Full Connectなネットワークは、もっと強い性質を持っている。

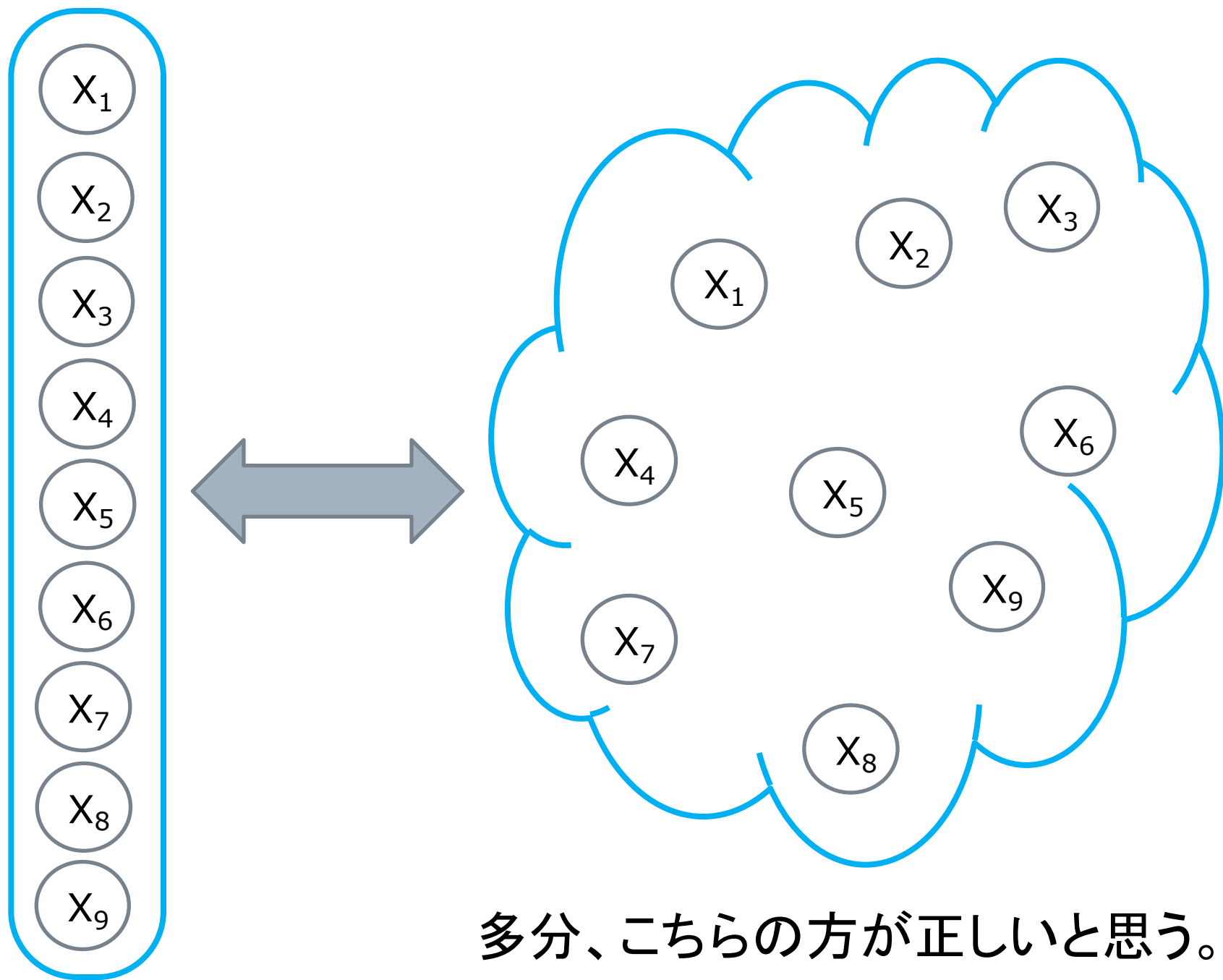




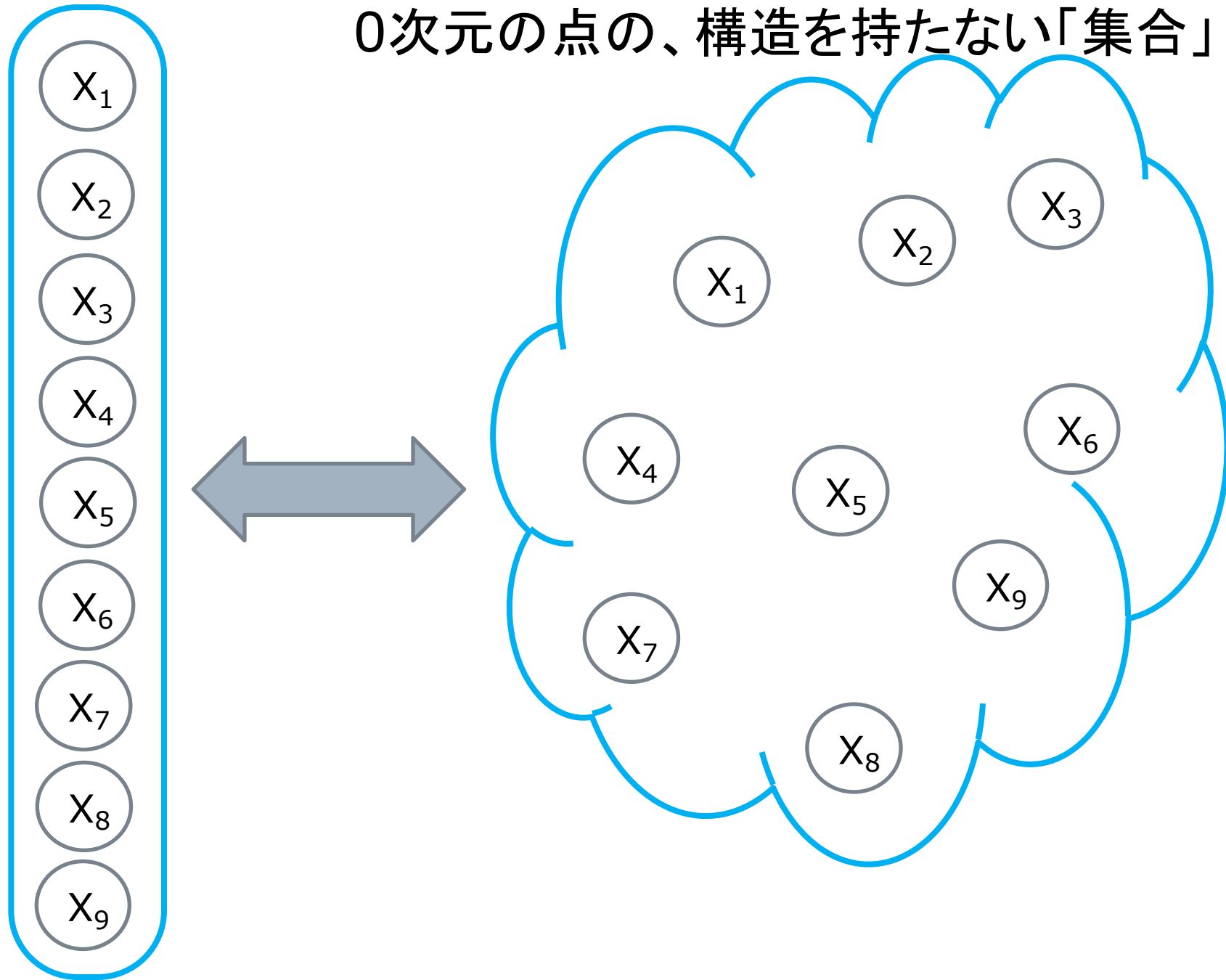
Full Connectのニューラル・ネットワークの入力層は、一次元の配列だと考えることが多い。

?

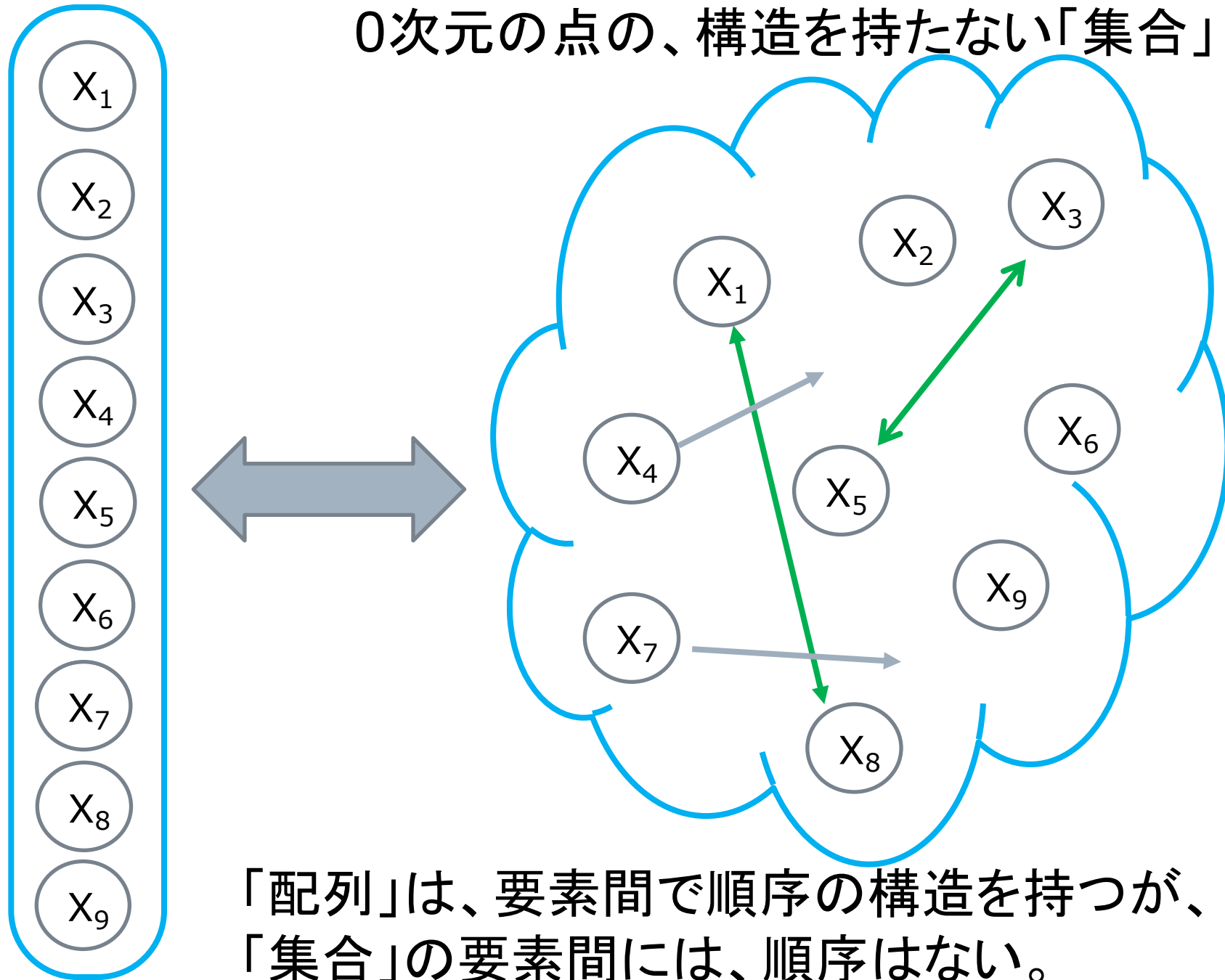
はたして、このイメージは、正しいのだろうか？



0次元の点の、構造を持たない「集合」

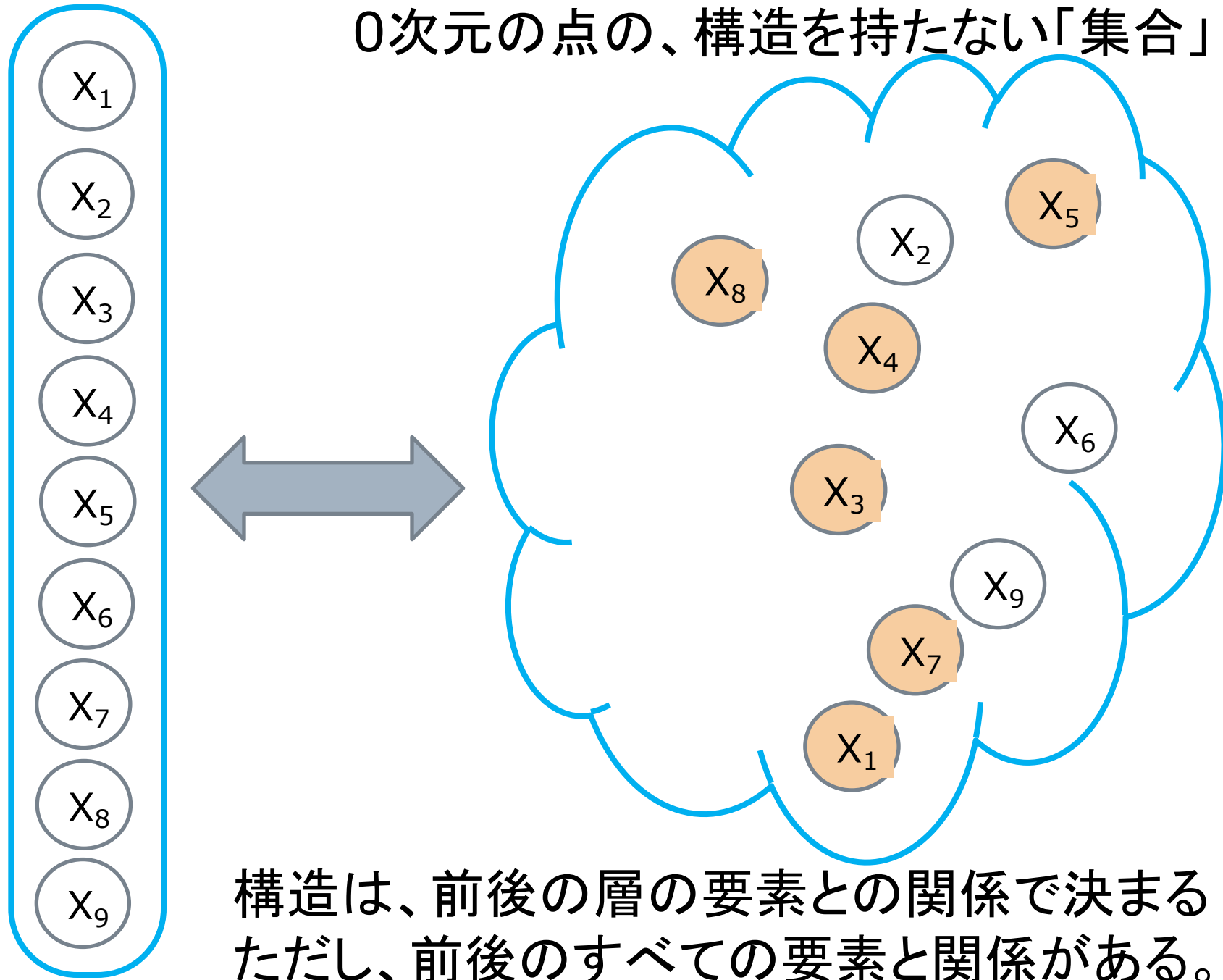


0次元の点の、構造を持たない「集合」

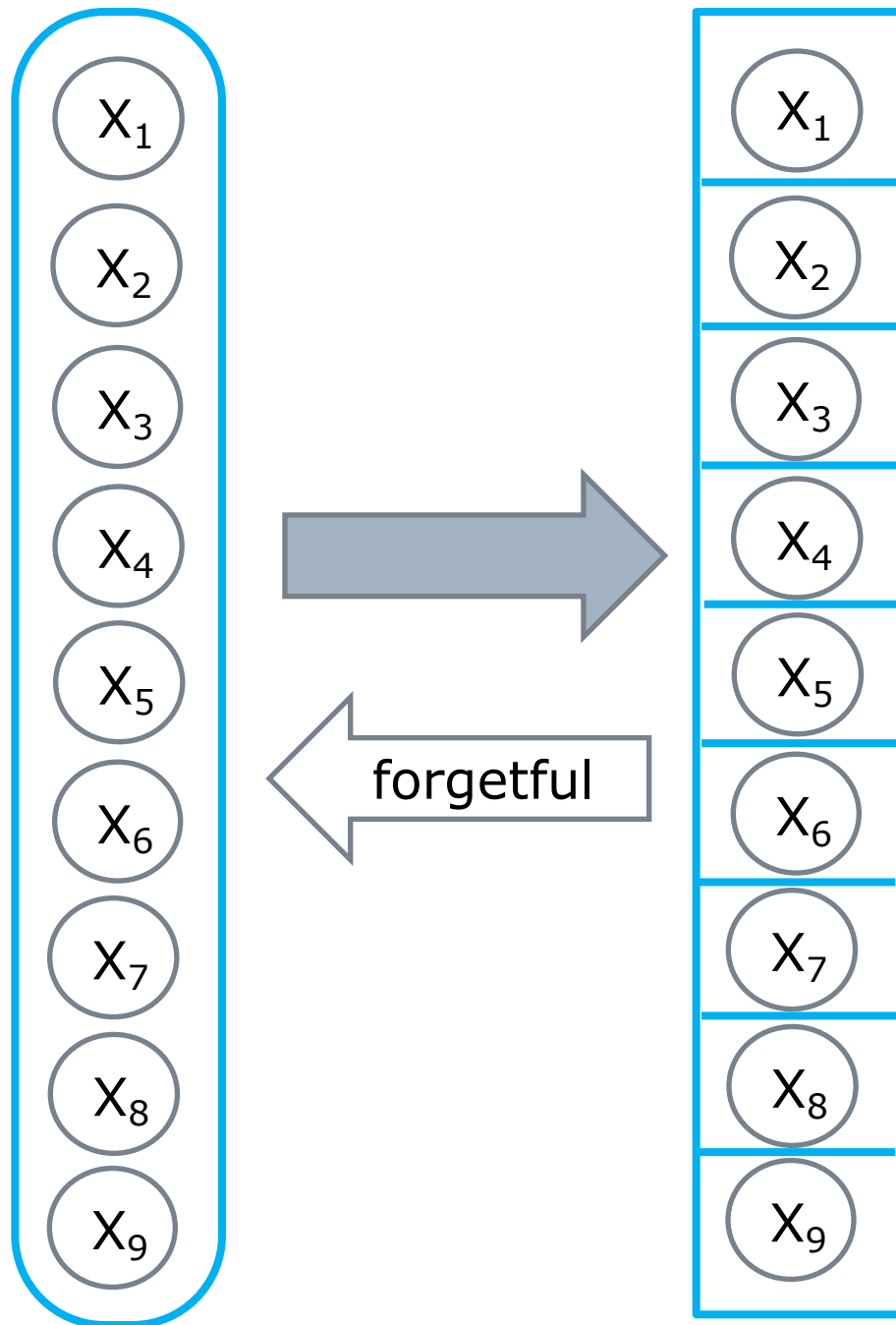


「配列」は、要素間で順序の構造を持つが、
「集合」の要素間には、順序はない。

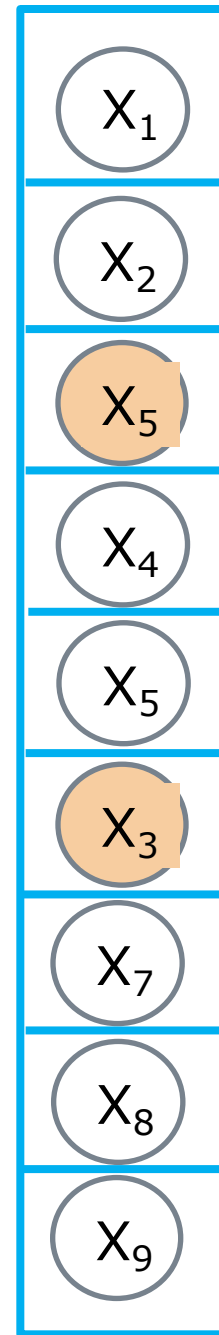
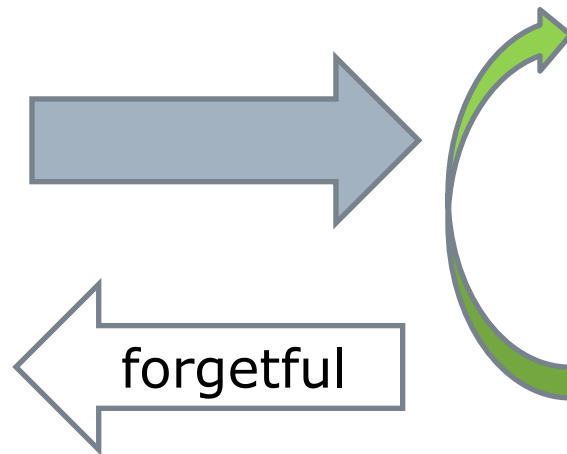
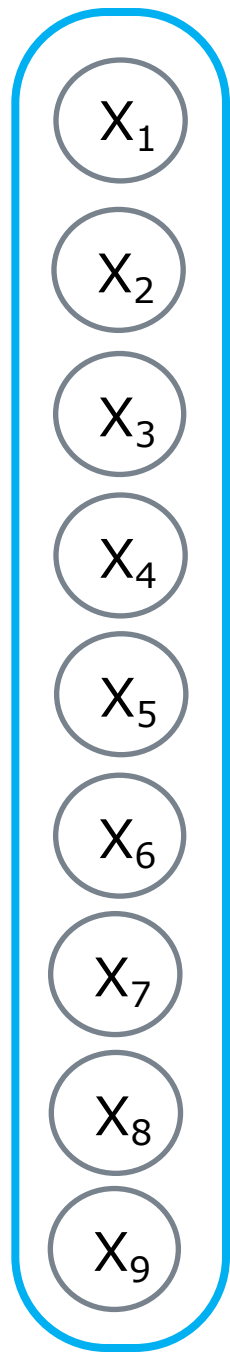
0次元の点の、構造を持たない「集合」



構造は、前後の層の要素との関係で決まる
ただし、前後のすべての要素と関係がある。



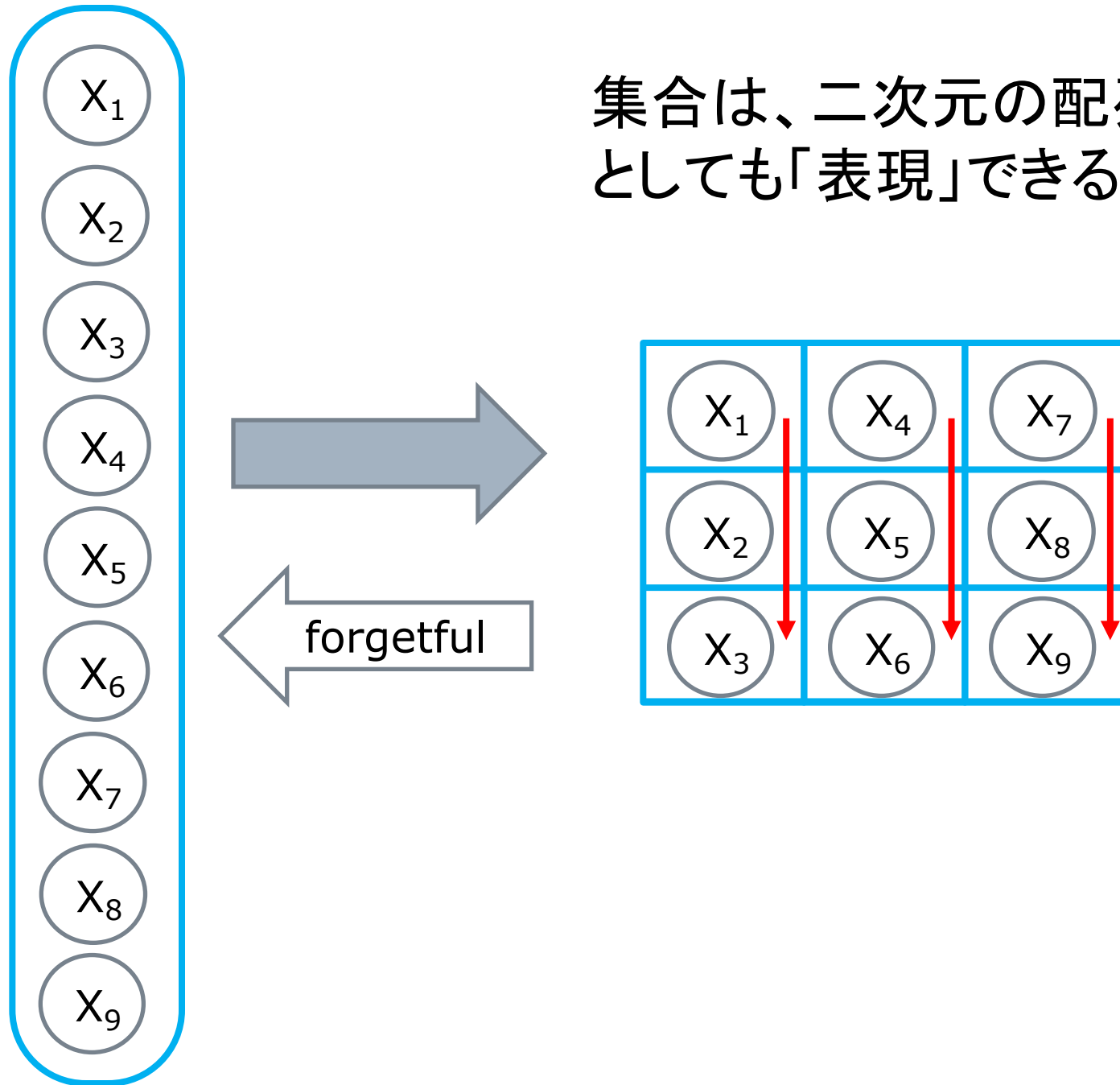
集合は、一次元の配列としても「表現」できる。ただし、集合は、配列に含まれる順序構造を、すべて「忘れた」もの。



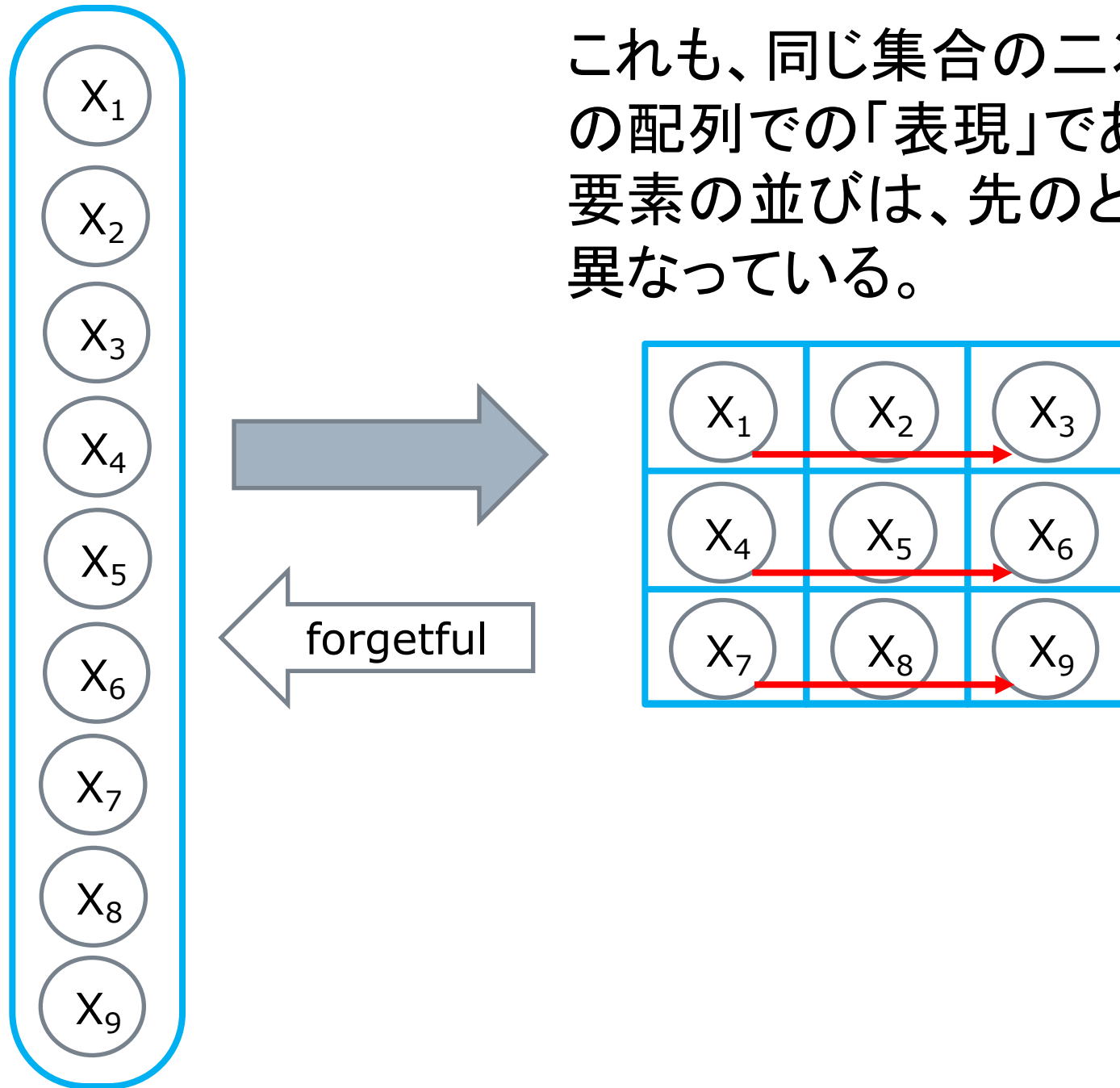
ある集合を、一次元の配列で表現したとしよう。その配列の要素を、一定の規則で置き換えてみよう。

この時、もとの配列と、置き換えられた配列は、同じ集合の「表現」である。

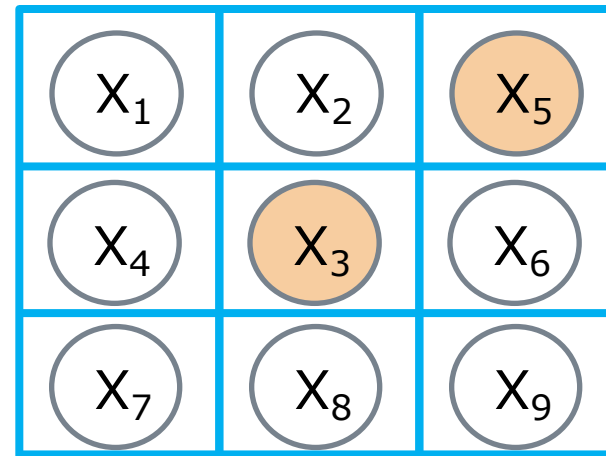
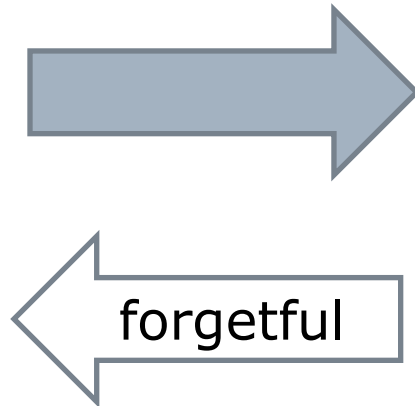
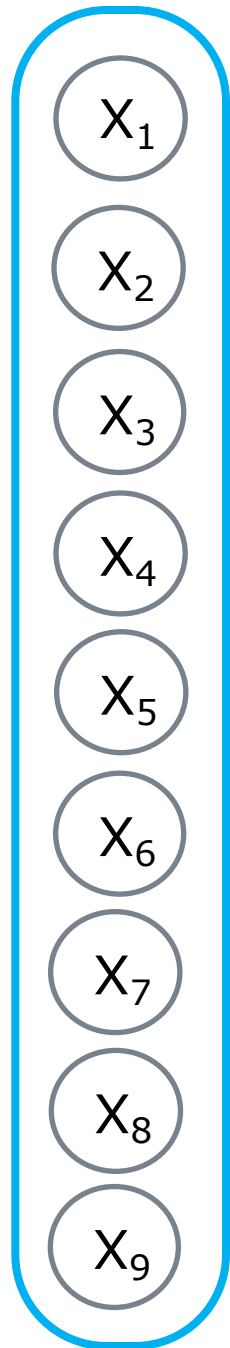
集合は、二次元の配列
としても「表現」できる。



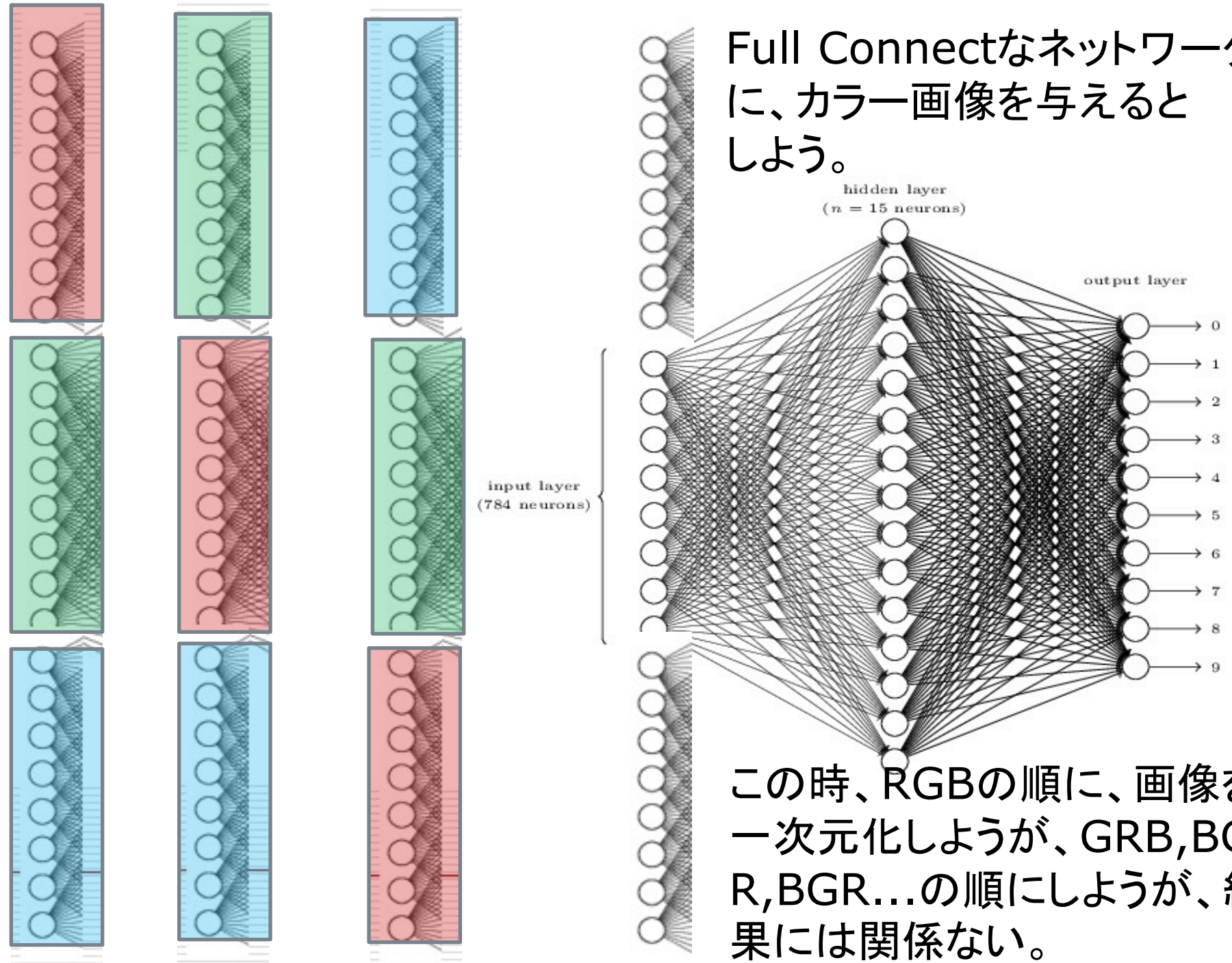
これも、同じ集合の二次元の配列での「表現」である。要素の並びは、先のととは異なっている。



これも、同じ集合の二次元の配列での「表現」である。要素の並びは、先のととは異なっている。



Full Connectなネットワークに、カラー画像を与えよう。



この時、RGBの順に、画像を一次元化しようが、GRB,BGR,R,BGR...の順にしようが、結果には関係ない。

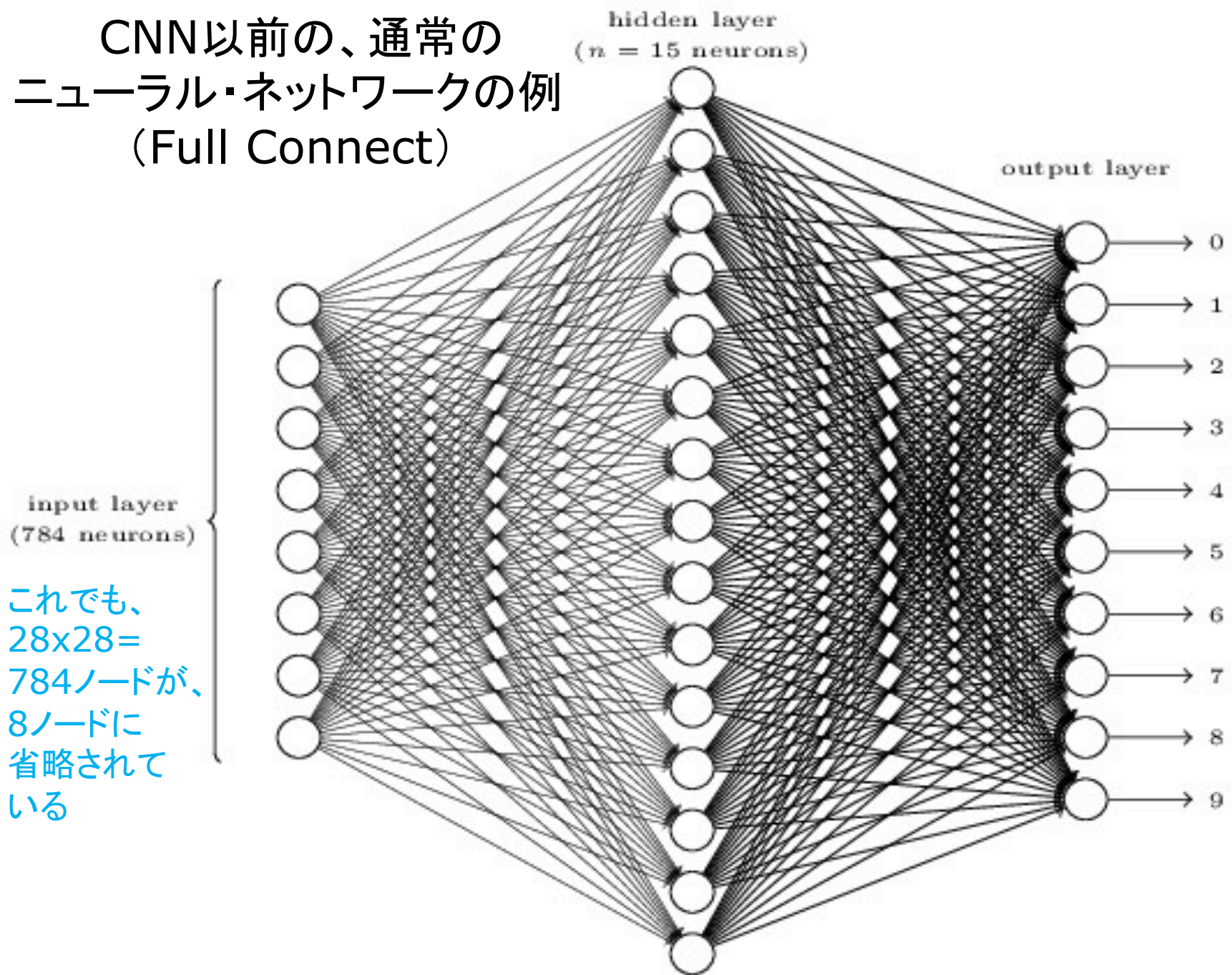
ニューラル・ネットワークの複雑さと、 そのシンプルな把握の為の工夫

ニューラル・ネットワークは、複雑なものである。それを図で示そうとすると、その複雑さがわかる。

ただ、そこには規則性がある。ネットワークを「グラフ」で表現し、その上を流れるデータを「テンソル」として捉え、それらを「モジュール」として組み合わせる。

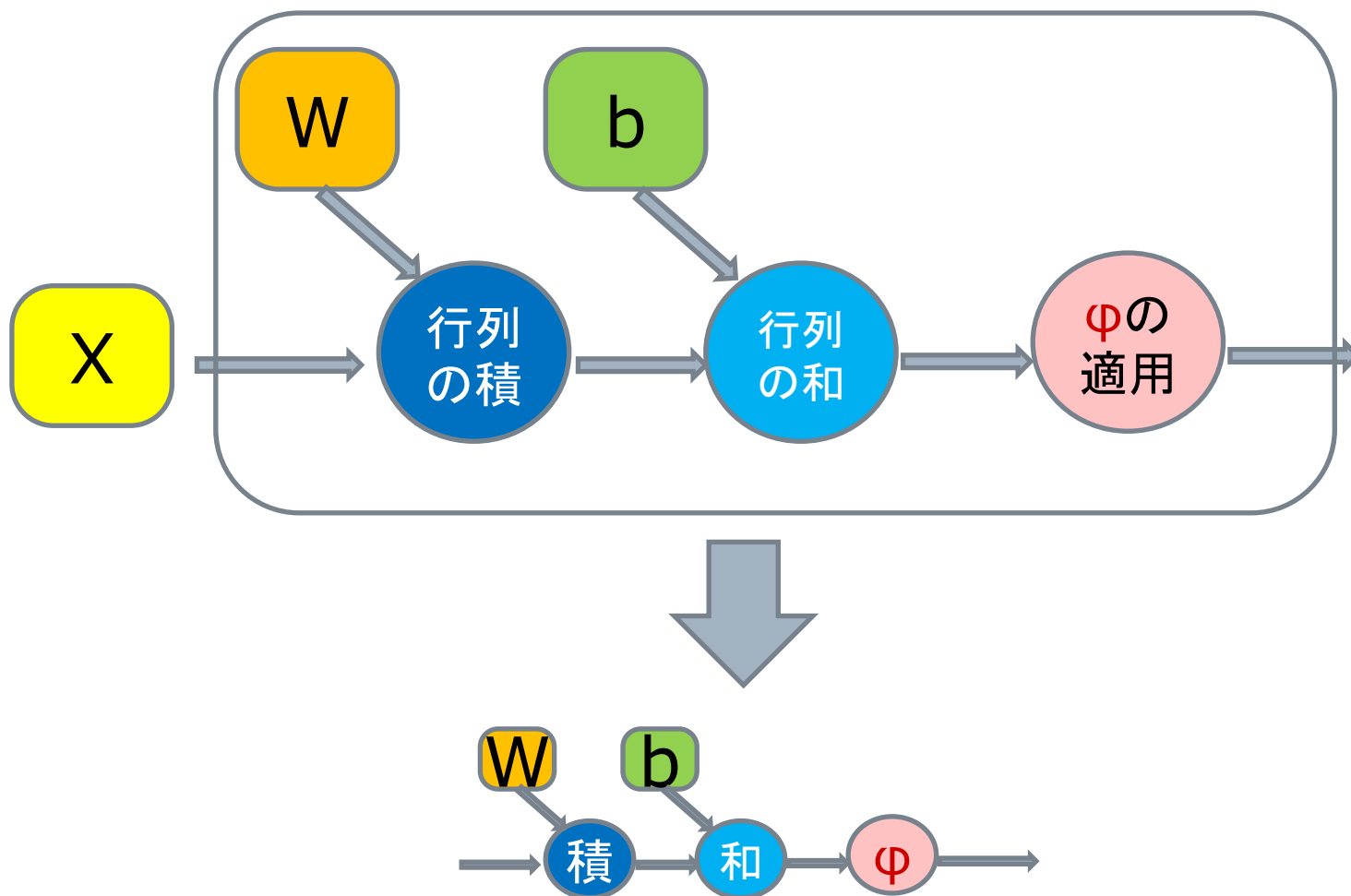
こうした把握は、ニューラル・ネットワークのプログラミングを容易なものにした。ただ、そのことは、Full Connectのネットワークを単純化するものではない。

CNN以前の、通常の ニューラル・ネットワークの例 (Full Connect)



ニューラル・ネットワークのモデュラー性

一つの「層」のグラフでの表現

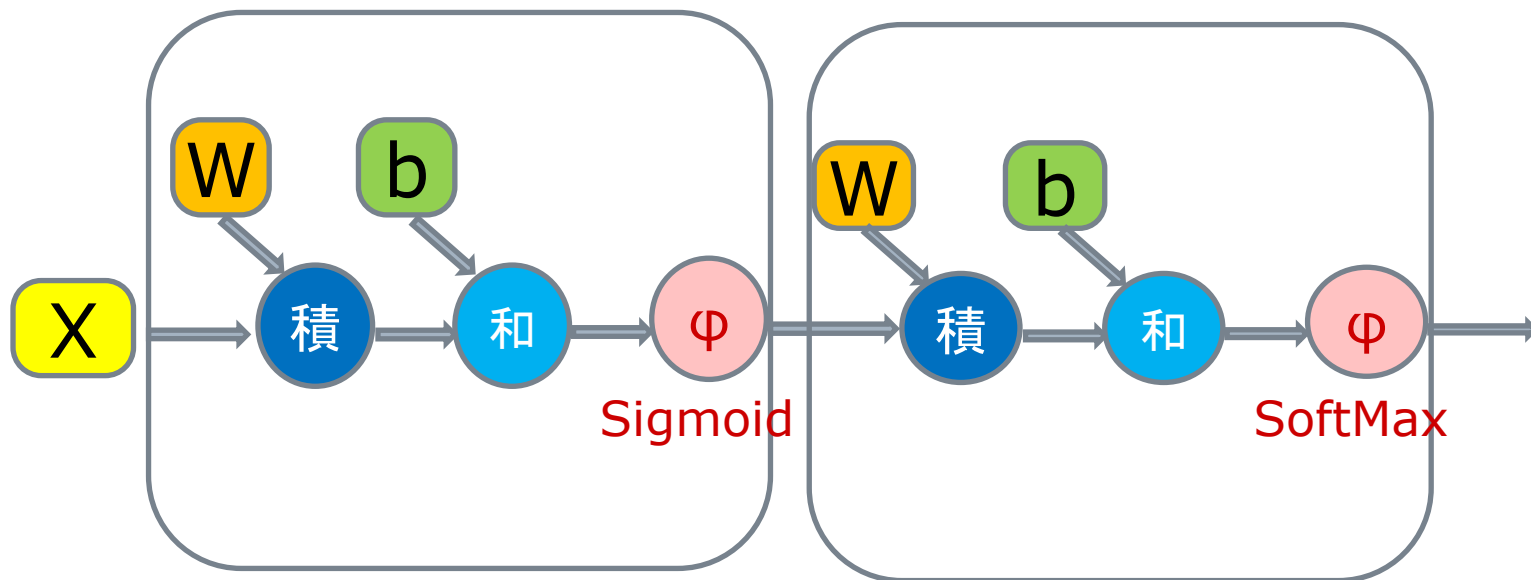


TensorFlowのグラフ

隠れ層の
入力: 784
784次元

隠れ層の
ニューロン数: 15
重み W_H : 784x15次元
バイアス b_H : 15次元

出力層の
ニューロン数: 10
重み W_O : 15x10次元
バイアス b_O : 10次元



入力層

隠れ層

出力層

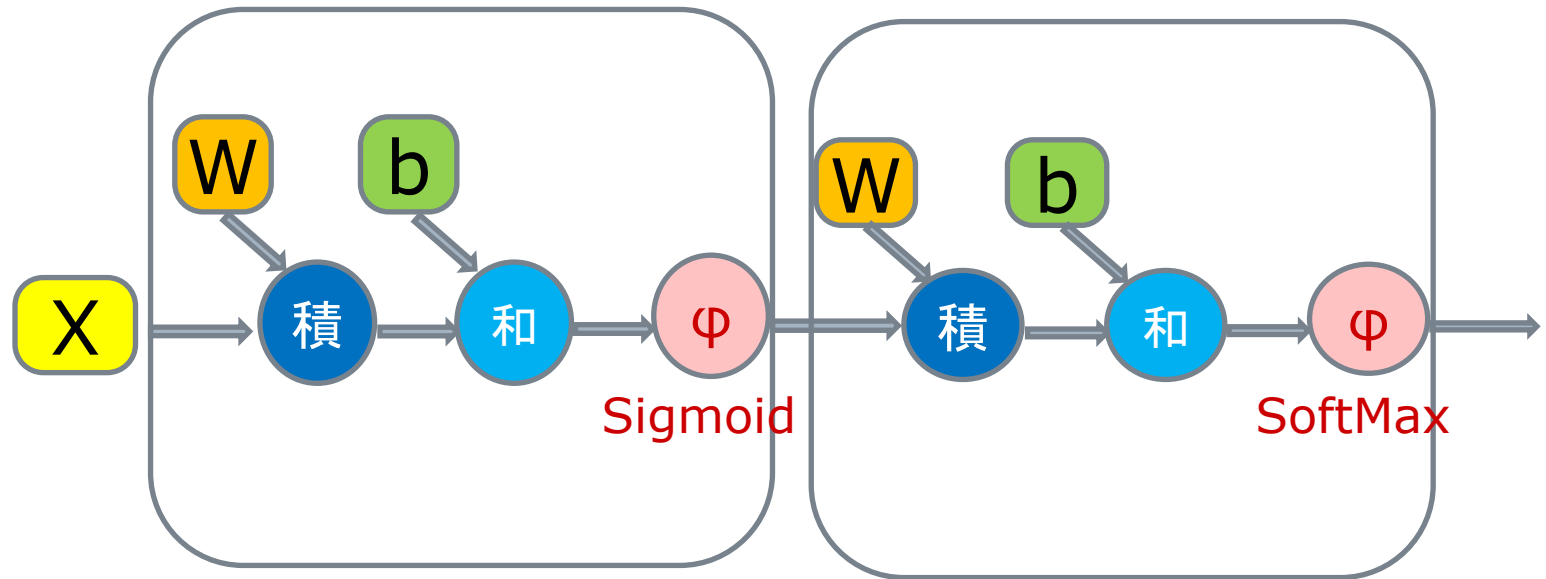
こっちのグラフの方が、ずっとわかりやすい！

TensorFlowのグラフ

隠れ層の
入力: 784
784次元

隠れ層の
ニューロン数: 15
重み W_H : 784x15次元
バイアス b_H : 15次元

出力層の
ニューロン数: 10
重み W_O : 15x10次元
バイアス b_O : 10次元



入力層

隠れ層

出力層

こっちのグラフの方が、ずっとわかりやすい！
本当に、それでいいのか？

CNN以前のネットワークの問題

CNN登場以前の、Full Connectなネットワークには、大きく言って、二つの問題がある。一つは、パラメーターの数の爆発であり、もう一つは、画像データの持つ局所的な特徴をうまく抽出できないということである。

CNN以前のネットワークの第一の問題

パラメーターの数の爆発

- 入力層に n 個のデータがあるとしよう。この時、最初の隠れ層の一つ一つのニューロンは、それぞれ、入力層のすべてのデータを受け取るので、 n 個のシナプスを持つ。それは、最初の隠れ層のニューロンが、 n 個の重みパラメーターを持つことを意味する。
 - この隠れ層のニューロンの数を m 個だとすると、最初の隠れ層の持つパラメーターの総数は、 $n \times m + n$ となる。
(最後の $+ n$ は、バイアスの個数である)
 - 実際の画像データで、この $n \times m$ が、どのくらいになるのか、コンピュータ上でよく使われる画像フォーマットで見てみよう。
-

よく使われる画像フォーマットの画素数

機械学習		総画素数	
MNIST	28x28 [GreyScale]	784x1	手書き数字 10 クラス
CIFAR-10	32x32	1024x3	CIFAR-10/100
ImageNet (ILSVRC-2010)	256x256	65,536x3	画像データ 1,000クラス
AlphaGO	19x19	361x1	
PC/TV			
VGA	640x480	307,200x3	DOS/V
XGA	1024x768	786,432x3	Windows3.0~XP
Full-WXGA	1366x768	1,049,088x3	低価格HDTVの主流
Full-HD	1920x1088	2,073,600x3	一般的なディスプレイ
4K	3840x2160	8,294,400x3	Xperia Z5 Premium
8K FUHD	7680x4320	33,177,600x3	

CNN以前のネットワークの第一の問題

パラメーターの数の爆発

- MNISTの入力データ数は $28 \times 28 = 784$ 、
CIFAR-10の入力データ数は $32 \times 32 \times 3 = 3,072$
CIFAR-10はカラー画像なので、 $\times 3$ になる、
ImageNetの入力データ数は、 $256 \times 256 \times 3 = 196,608$
Full Connectなネットワークでは隠れ層の一個のニューロンについて、これだけのパラメーターが必要になる。
- もうひとつ注意してほしいのは、これらの機械学習でよく用いられる画像データは、コンピュータの世界で一般的な画像データと比較すると、必ずしも大きなものではないということである。それは、25年前のDOS/Vの画像データより小さい。
- ちなみに、AlphaGOの入力データは、 $19 \times 19 \times 1$ である。

Example dataset: **CIFAR-10**

10 labels

50,000 training images, each image is tiny: 32x32

10,000 test images.

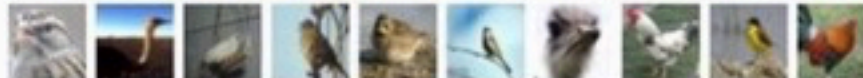
airplane



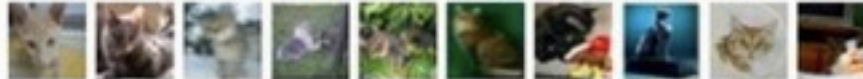
automobile



bird



cat



deer



dog



frog



horse



ship



truck



ImageNet 1000 Categories x AlexNet



mite

container ship

motor scooter

leopard

mite	container ship	motor scooter	leopard
black widow	lifeboat	go-kart	jaguar
cockroach	amphibian	moped	cheetah
tick	fireboat	bumper car	snow leopard
starfish	drilling platform	golfcart	egyptian cat



grille

mushroom

cherry

Madagascar cat

convertible	agaric	dalmatian	squirrel monkey
grille	mushroom	grape	spider monkey
pickup	jelly fungus	elderberry	titi
beach wagon	gill fungus	fordshire bullterrier	indri
fire engine	dead-man's-fingers	currant	howler monkey



ImageNet (ILSVRC-2010)



Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 6 - 83

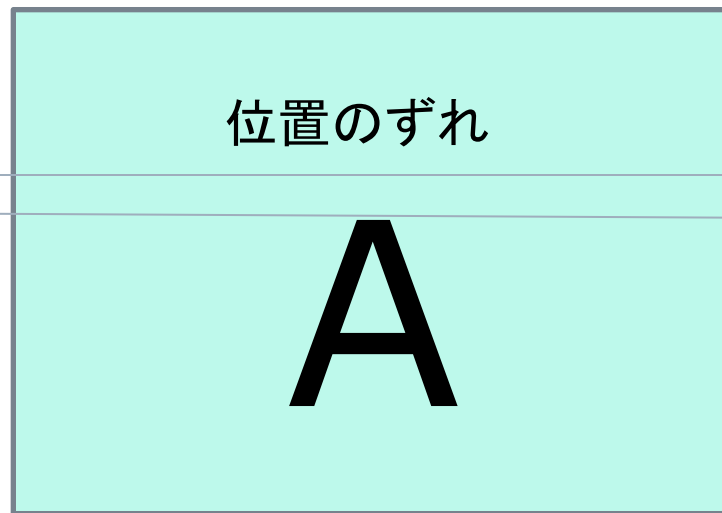
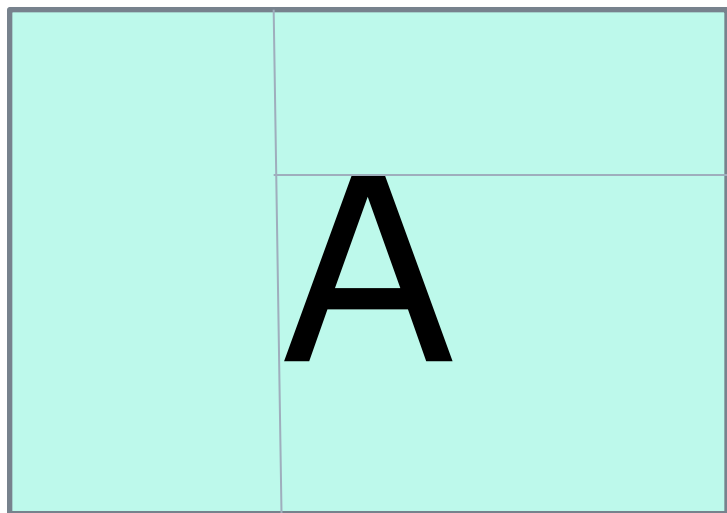
25 Jan 2016

14,197,122 images, 21,841 synsets indexed

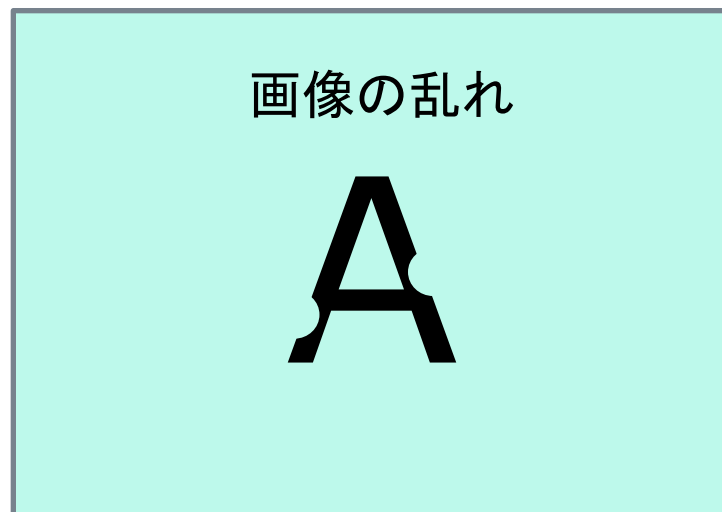
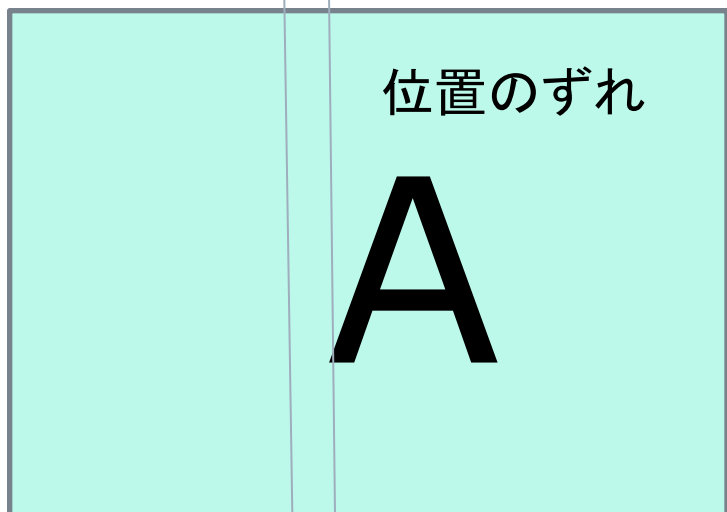
CNN以前のネットワークの第二の問題

画像データの特徴を捉えているか？

- ❑ Full Connectなネットワークでは、入力層の全てのノードが、次の層の全てのノードと接続しているので、入力層上でのデータの並びは、大きな意味を持たない。
- ❑ ただ、画像データでは、一つ一つのピクセルの位置情報とその並びは、意味を持つ。同時に、画像認識という視点から考えれば、ピクセルの微細な位置のずれや画像の乱れは、画像の認識にとっては、大きな意味を持たない。こうした、相反する要求を、同時に満たすことが、求められることになる。
- ❑ また、RGBで表された三枚の画像のデータには、対応する箇所で、強い相関がある。同様に、画像上の特徴的なパターンは、データ内部の局所的な強い相関として捉えることができる。



画像の微細な位置のズレや乱れ



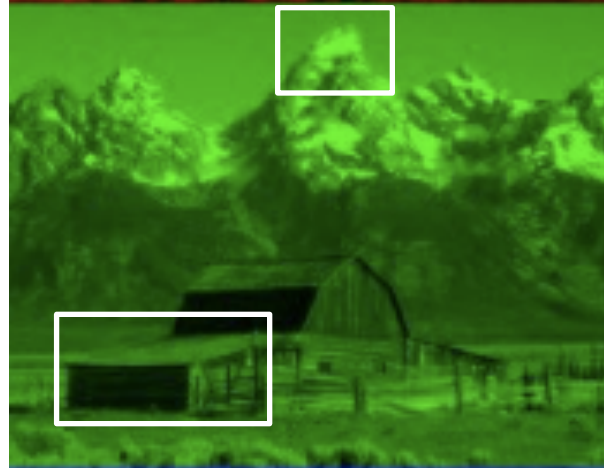
カラー画像のRGB表現



R



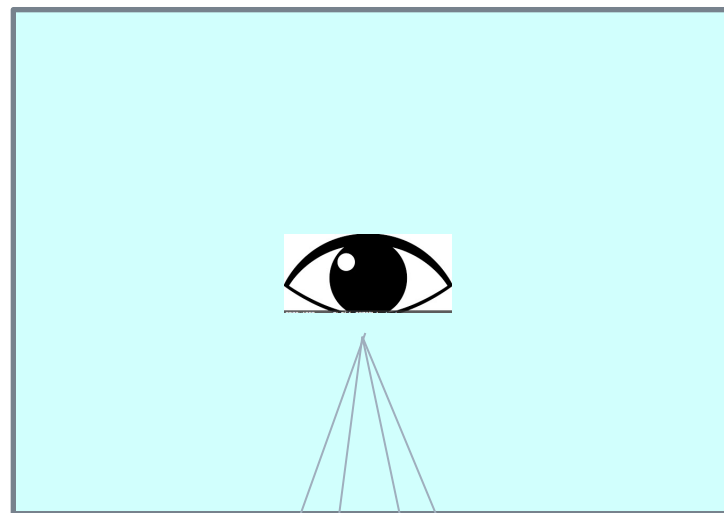
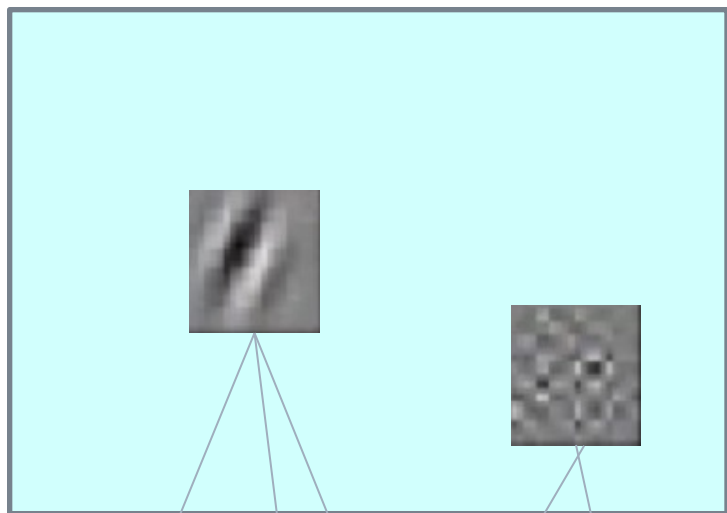
G



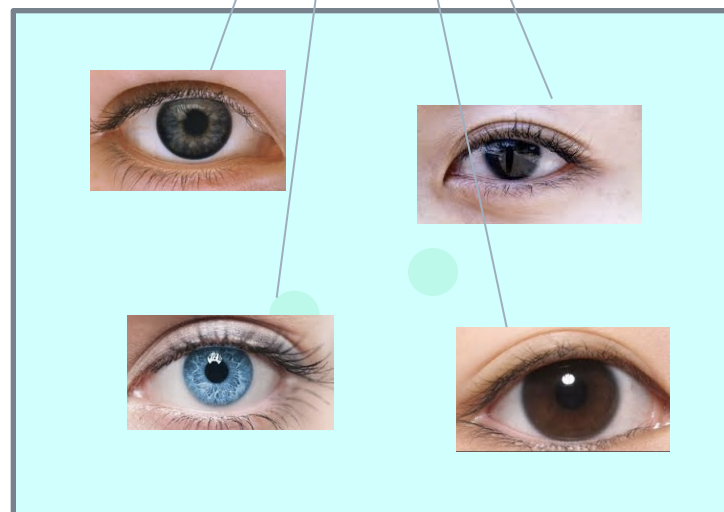
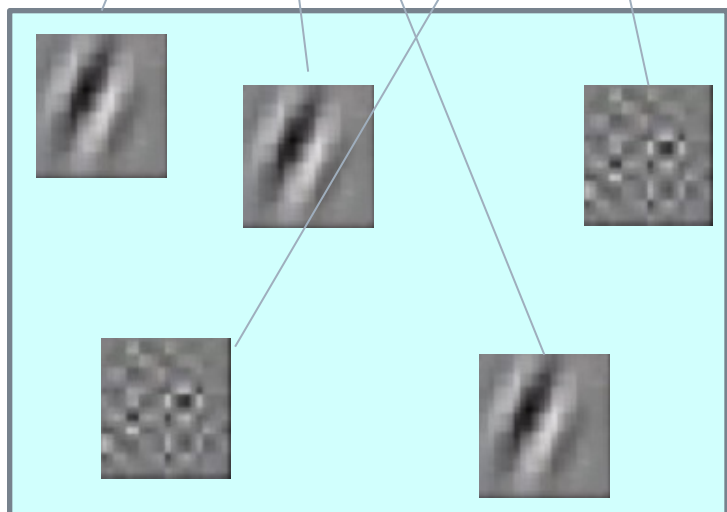
B



三つの画像のピクセルには相関がある



特徴的なパターン



CNNのアーキテクチャー

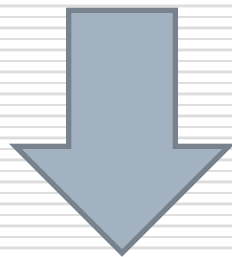
CNNは、これまで見たように、それ以前のFull Connectなネットワークが抱えていた問題を解決しようという試みから生まれた。CNNは、問題に、どのように対応しようとしたのだろうか？ ここでは、それを見てみよう。

Full Connectなネットワークの問題

- パラメーターの数の爆発 → 完全二部グラフの見直し
 - データの局所的な情報の抽出 → 入力データに、明確な構造と位置情報を与える。完全二部グラフの見直し
 - 微細なシフトやゆがみの捨象 → 解像度を下げる
640x480 → 320x240 → 160x120 → 80x60
後述する Pooling層は、そうした役割を持つ。
-

Full Connectなネットワークの問題

- パラメーターの数の爆発 → 完全二部グラフの見直し
- データの局所的な情報の抽出 → 入力データに、明確な構造と位置情報を与える。完全二部グラフの見直し
- 微細なシフトやゆがみの捨象 → 解像度を下げる

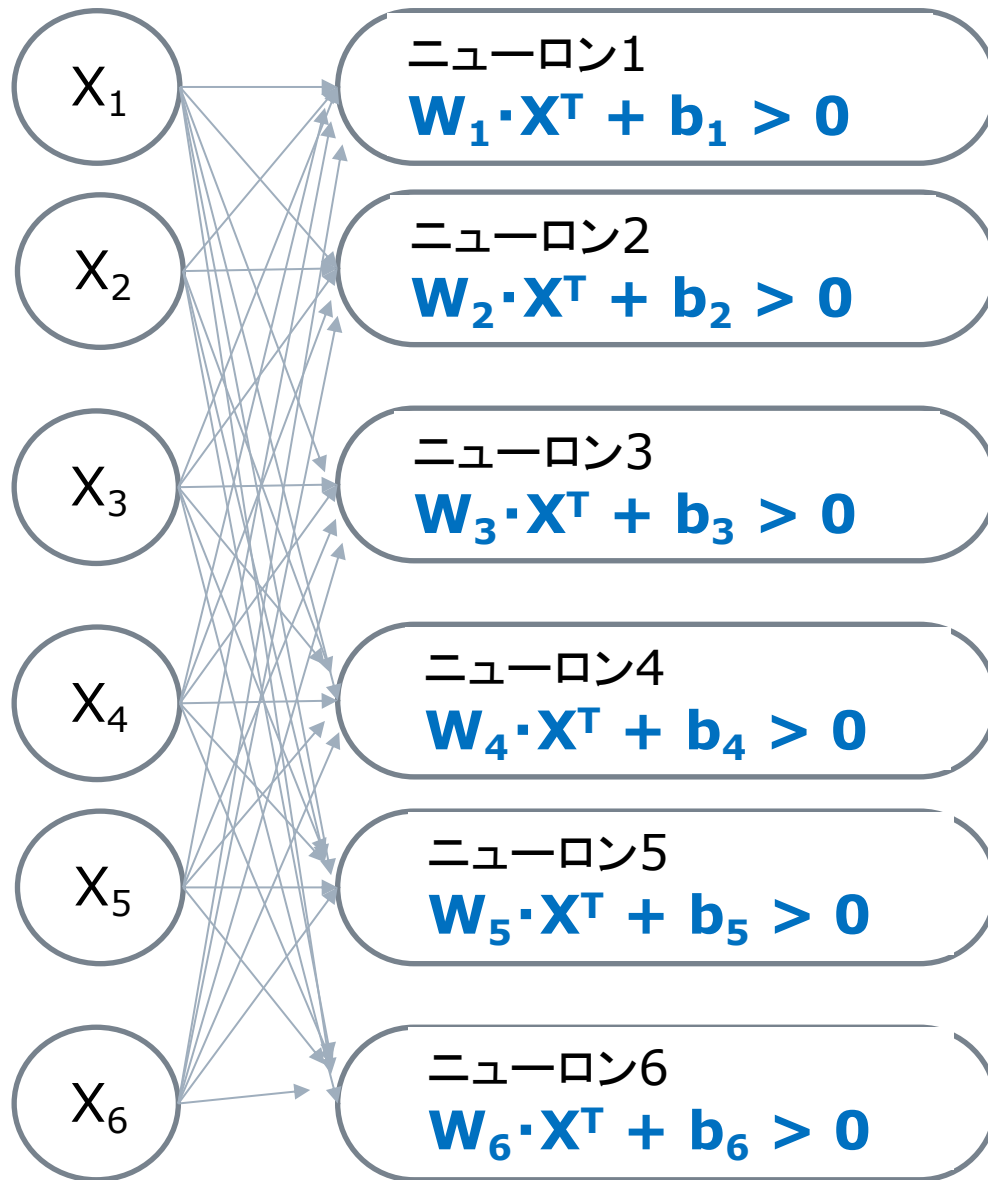


- Full Connectな、ニューラル・ネットを、あらためて、チューンする
-

ニューラル・ネットをチューンする

- 入力データを、Full Connect = 完全二部グラフのように、「集合」ではなく、「幅」と「高さ」（と「深さ(RGB)」）を持った、ピクセルの順序を持った「配列」として表現する。そうした位置情報によって、「近い」ピクセル間の、「局所的な相関関係」を表現できる。
 - 次層のニューロンは、Full Connect = 完全二部グラフのように、入力層のすべてのノードのデータを受け入れるのではなく、隣接したノードからなる、限られた領域のノードのデータのみを受け入れるようにする。（「局所的な受容野」） それは、パラメーターの数を大きく減らす。
-

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$



Full Connect な場合

$$\text{値: } W_{11}X_1 + W_{12}X_2 + W_{13}X_3 + W_{14}X_4 + W_{15}X_5 + W_{16}X_6 + b_1$$

$$\text{値: } W_{21}X_1 + W_{22}X_2 + W_{23}X_3 + W_{24}X_4 + W_{25}X_5 + W_{26}X_6 + b_2$$

$$\text{値: } W_{31}X_1 + W_{32}X_2 + W_{33}X_3 + W_{34}X_4 + W_{35}X_5 + W_{36}X_6 + b_3$$

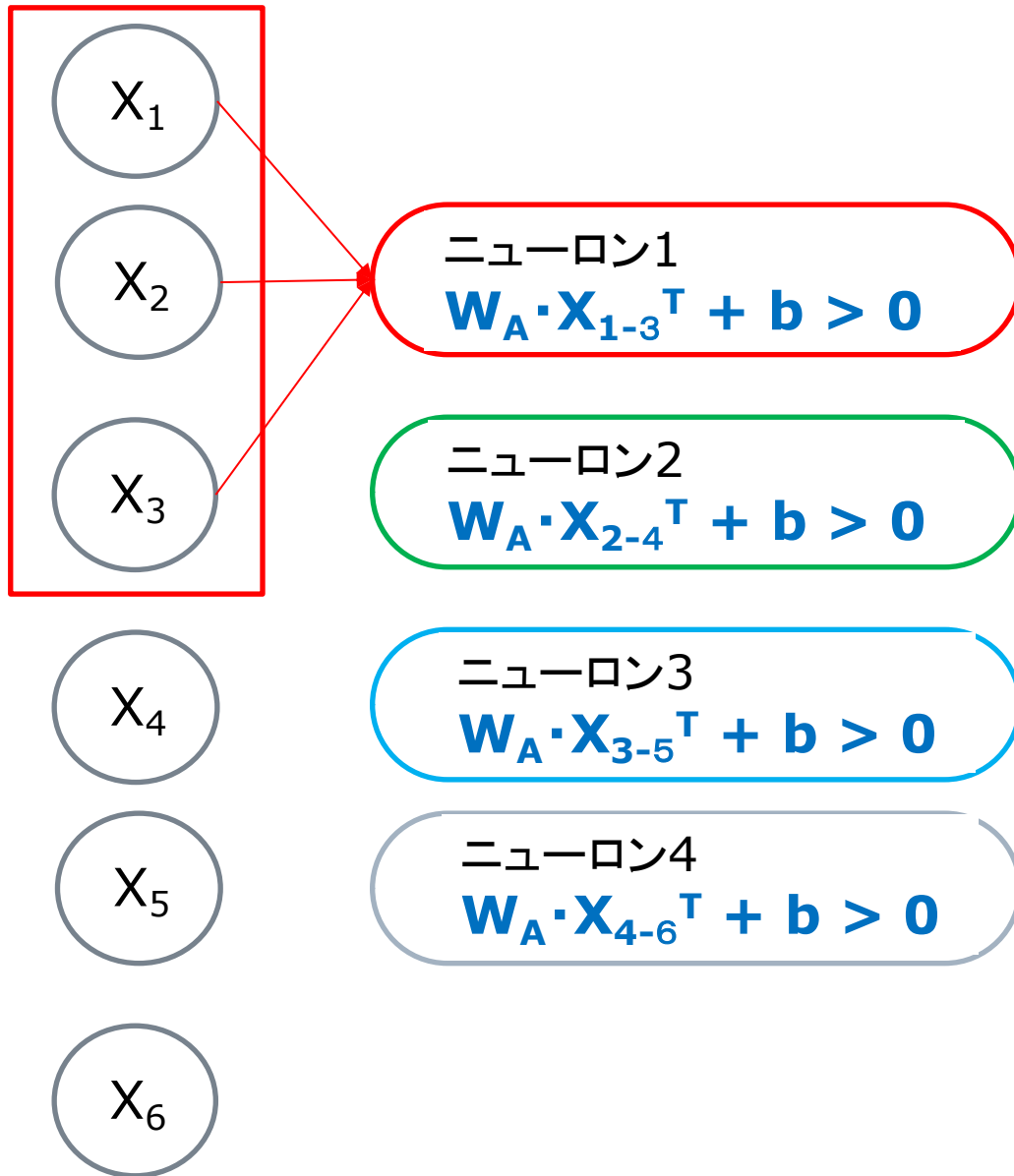
$$\text{値: } W_{41}X_1 + W_{42}X_2 + W_{43}X_3 + W_{44}X_4 + W_{45}X_5 + W_{46}X_6 + b_4$$

$$\text{値: } W_{51}X_1 + W_{52}X_2 + W_{53}X_3 + W_{54}X_4 + W_{55}X_5 + W_{56}X_6 + b_5$$

$$\text{値: } W_{61}X_1 + W_{62}X_2 + W_{63}X_3 + W_{64}X_4 + W_{65}X_5 + W_{66}X_6 + b_6$$

パラメーターの数は、 $6 \times 6 + 6 = 42$

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$

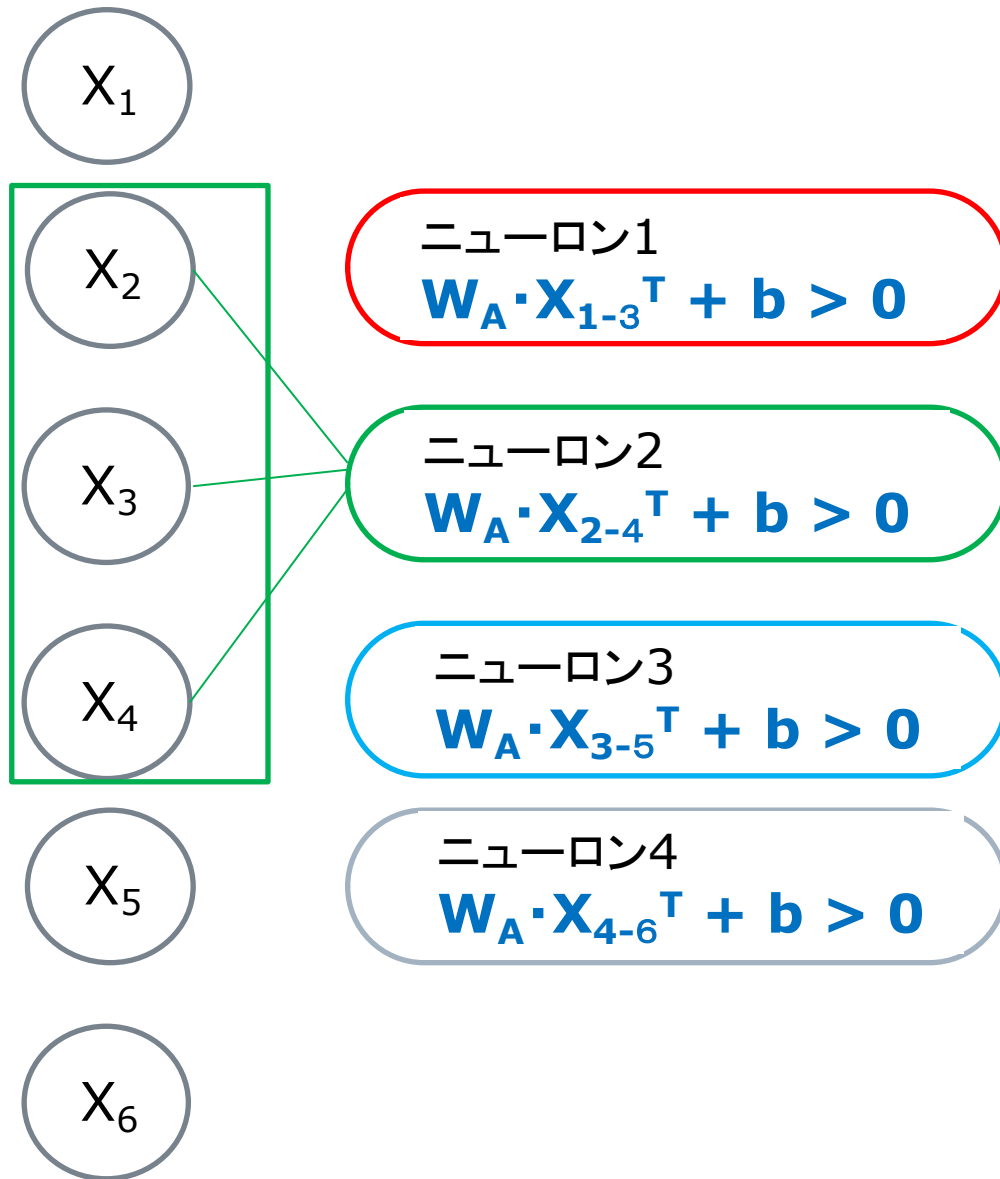


局所的な受容野と フィルター

ニューロン1は、入力層の X_1, X_2, X_3 の三つからしかデータを受け取らない。その他のニューロンも、同様に、三つのデータしか受け取らない。

これらのニューロンは、入力データの小部分を変換する局所的な受容野、局所的なフィルターとして機能する。

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$



局所的な受容野と フィルター

ニューロン2は、入力層の X_2, X_3, X_4 の三つからしかデータを受け取らない。その他のニューロンも、同様に、三つのデータしか受け取らない。

これらのニューロンは、入力データの小部分を変換する局所的な受容野、局所的なフィルターとして機能する。

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$

X_1

X_2

X_3

X_4

X_5

X_6

ニューロン1

$$\mathbf{W}_A \cdot \mathbf{X}_{1-3}^T + b > 0$$

ニューロン2

$$\mathbf{W}_A \cdot \mathbf{X}_{2-4}^T + b > 0$$

ニューロン3

$$\mathbf{W}_A \cdot \mathbf{X}_{3-5}^T + b > 0$$

ニューロン4

$$\mathbf{W}_A \cdot \mathbf{X}_{4-6}^T + b > 0$$

局所的な受容野と フィルター

ニューロン3は、入力層の X_3, X_4, X_5 の三つからしかデータを受け取らない。その他のニューロンも、同様に、三つのデータしか受け取らない。

これらのニューロンは、入力データの小部分を変換する局所的な受容野、局所的なフィルターとして機能する。

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$

X_1

X_2

X_3

X_4

X_5

X_6

ニューロン1

$$\mathbf{W}_A \cdot \mathbf{X}_{1-3}^T + b > 0$$

ニューロン2

$$\mathbf{W}_A \cdot \mathbf{X}_{2-4}^T + b > 0$$

ニューロン3

$$\mathbf{W}_A \cdot \mathbf{X}_{3-5}^T + b > 0$$

ニューロン4

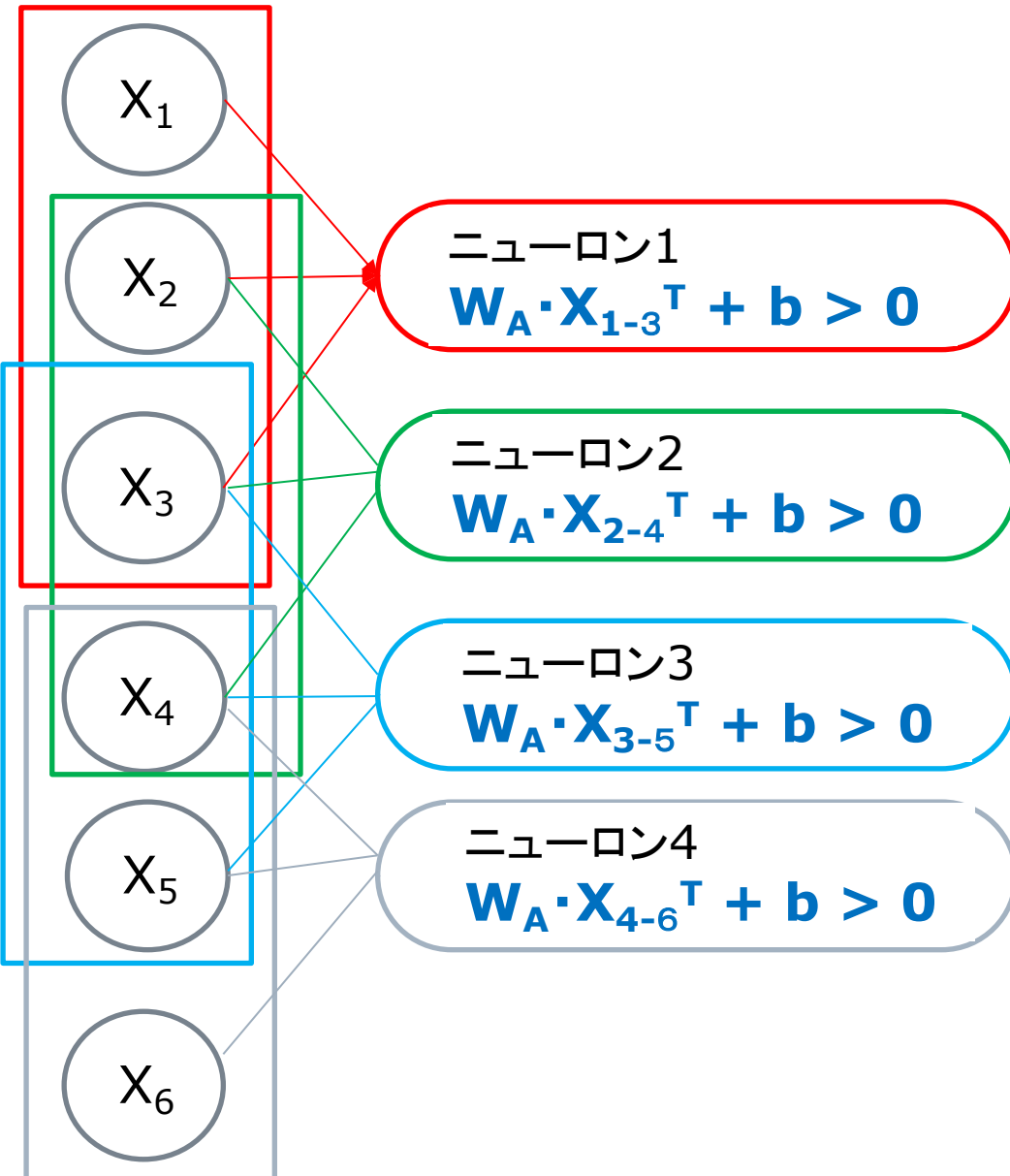
$$\mathbf{W}_A \cdot \mathbf{X}_{4-6}^T + b > 0$$

局所的な受容野と フィルター

ニューロン4は、入力層の X_4, X_5, X_6 の三つからしかデータを受け取らない。その他のニューロンも、同様に、三つのデータしか受け取らない。

これらのニューロンは、入力データの小部分を変換する局所的な受容野、局所的なフィルターとして機能する。

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$

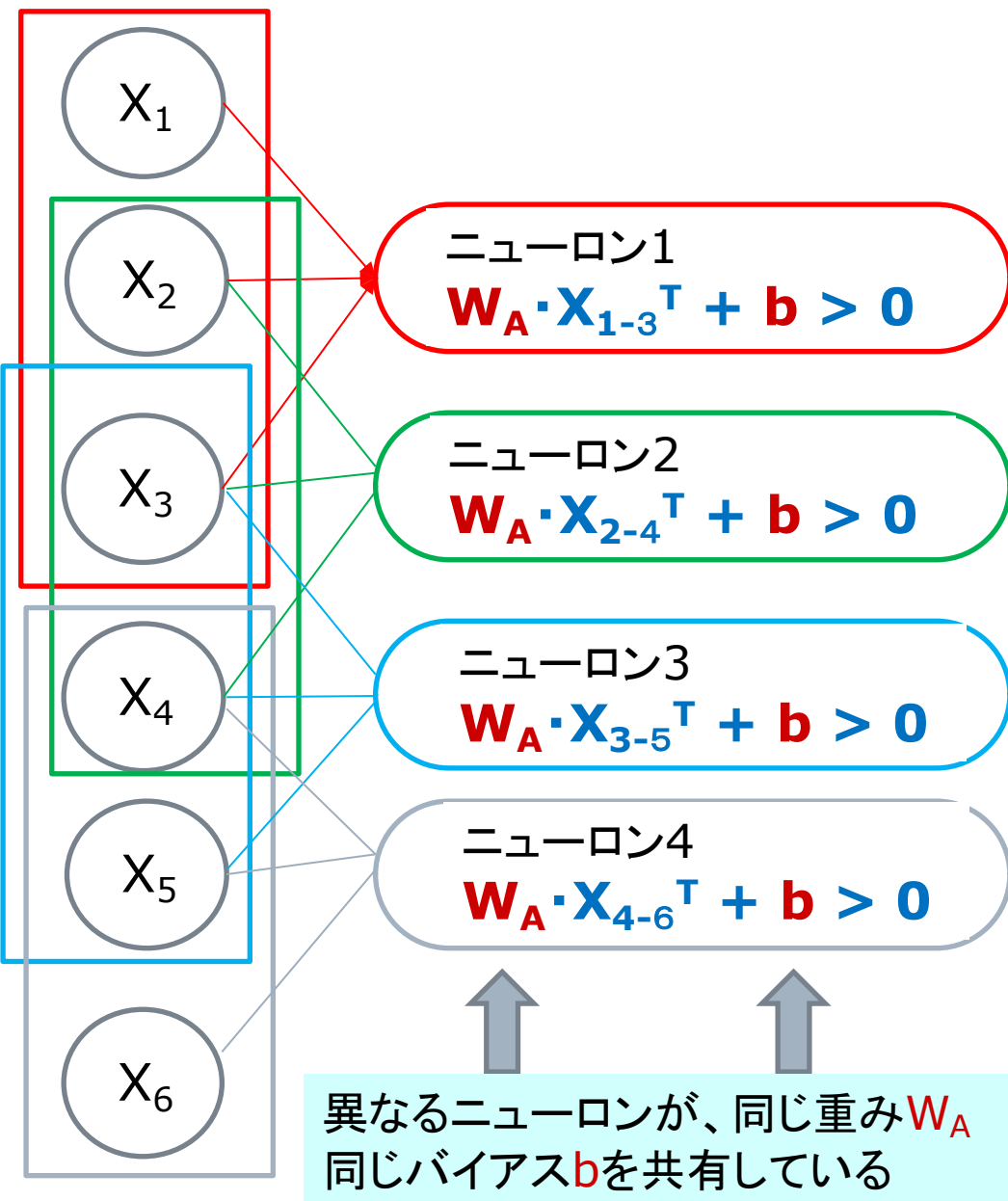


局所的な受容野とフィルター

ニューロン1は、入力層の X_1, X_2, X_3 の三つからしかデータを受け取らない。その他のニューロンも、同様に、三つのデータしか受け取らない。

これらのニューロンは、入力データの小部分を変換する局所的な受容野、局所的なフィルターとして機能する。

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$



パラメーターの共有

より重要なことは、ここでは、すべてのニューロンが、同じパラメーター(重み W_A とバイアス b)を共有していることである。

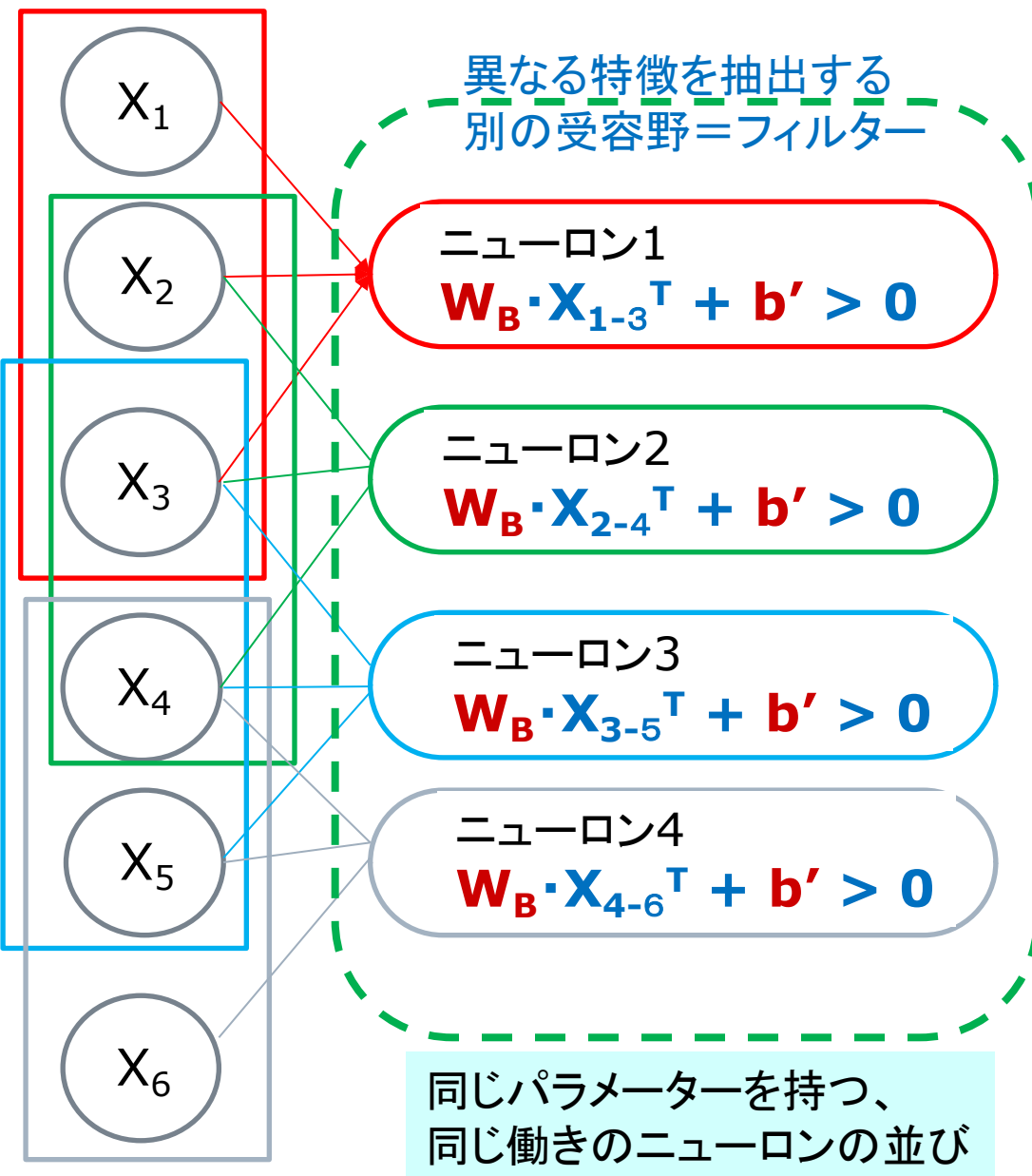
これらのニューロンは、同じ働きをする同じものであるとみなしていい。それは、同一の局所的な特徴を検出していると考えることができる。

W_A は、三つの入力を受け取るので、次の形をしている。

$$W_A = [W_{A1}, W_{A2}, W_{A3}]$$

パラメーターの数は、 $3+1=4$

入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$



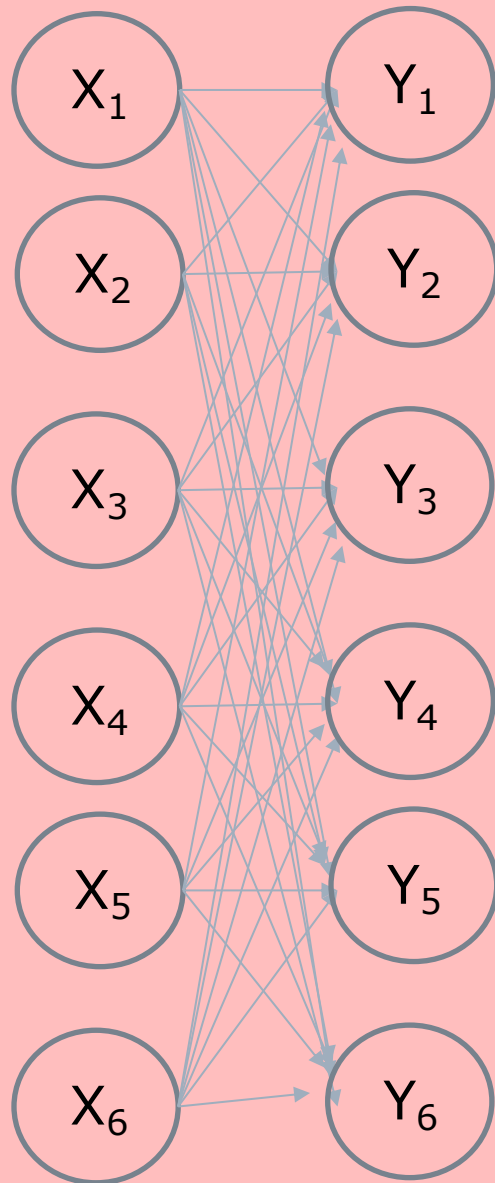
パラメーターの共有と異なるフィルターの追加

同じ入力 X に対して、異なるパラメーター $\mathbf{W}_B, b'$ を持つ一群のニューロンを考えることができる。ここでも、すべてのニューロンが、同じパラメーターを共有している。これらのニューロンは、先のニューロン群とは異なる局所的な特徴を検出していると考えることができる。

$\mathbf{W}_A, \mathbf{W}_B$ と同様に、 $\mathbf{W}_C, \mathbf{W}_D, \dots$ と、特徴検出のニューロン群を追加しても、パラメータの数は、大きくは増えない。

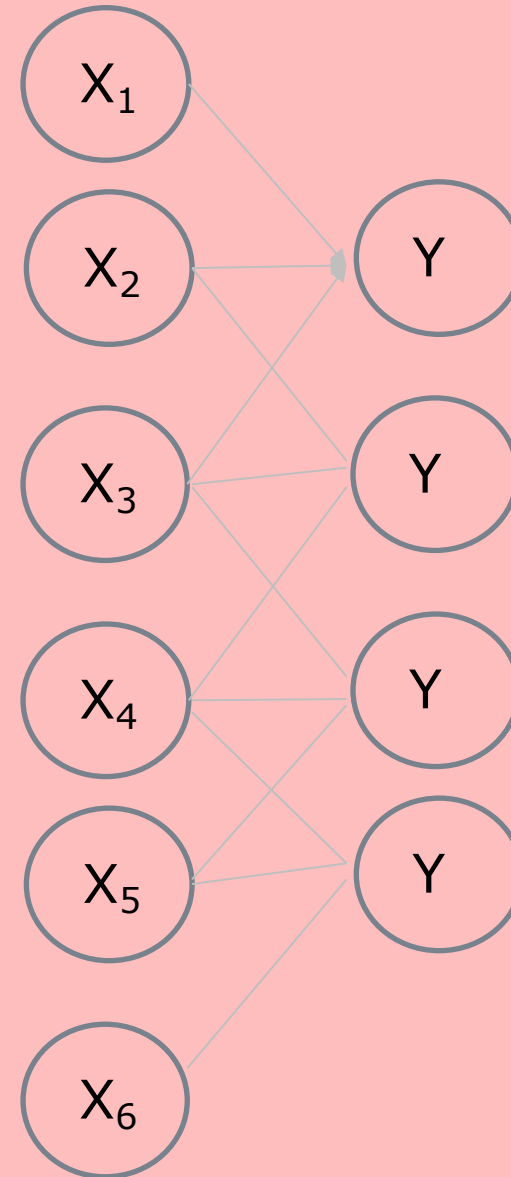
パラメーターの数は、4のn倍

Full Connect



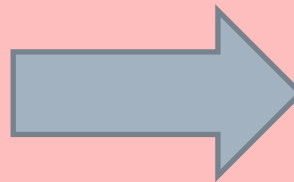
パラメーターの数は、 $6 \times 6 + 6 = 42$

CNN



パラメーターの数は、 $3 + 1 = 4$

接続の
局所化



パラメータ
の共有

Christpher Olahの“Conv Nets: A Modular Perspective”

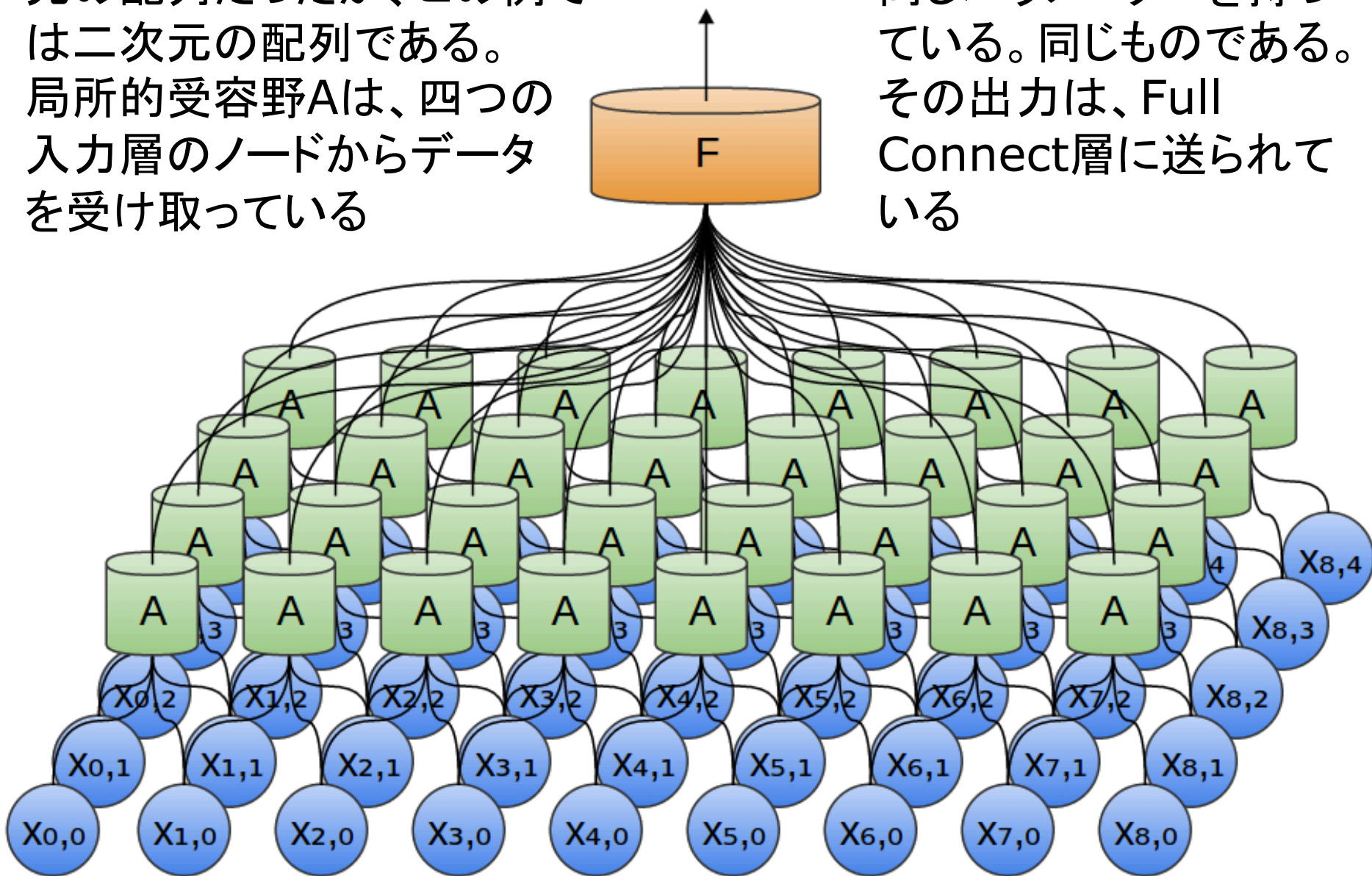
<http://goo.gl/EuPGgP> から

その最も基本的なところでは、コンボリューショナル・ニューラル・ネットワークは、同じニューロンの沢山の同一のコピーを利用するニューラル・ネットワークの一種だと考えることができる。このアプローチは、学習されるべき実際のパラメーターの数をかなり小さなものに抑えながら、パラメーターの値がニューロンがいかに関わるかを記述するのだが、ネットワークが多くニューロンを持ち、計算的には大きなモデルを表現することを可能にする。

同一のニューロンの複数のコピーを持つという、このトリックは、大まかに言えば、数学やコンピューター・サイエンスでの関数の抽象に似たものである。プログラミングをするとき、我々は関数を一度だけ書いて、それを多くの場所で利用する。同じコードを違った場所で何百回も書かないことは、プログラムを速くし、バグを少なくすることにもなる。同様に、コンボリューショナル・ニューラル・ネットワークでは、ニューロンが一度学習すれば、それを多くの場所で利用することができる。そうして、モデルの学習は容易になり、エラーも減らせることになる。

先の例では、入力層は一次元の配列だったが、この例では二次元の配列である。局所的受容野Aは、四つの入力層のノードからデータを受け取っている

局所的受容野Aは、全て同じパラメーターを持っている。同じものである。その出力は、Full Connect層に送られている

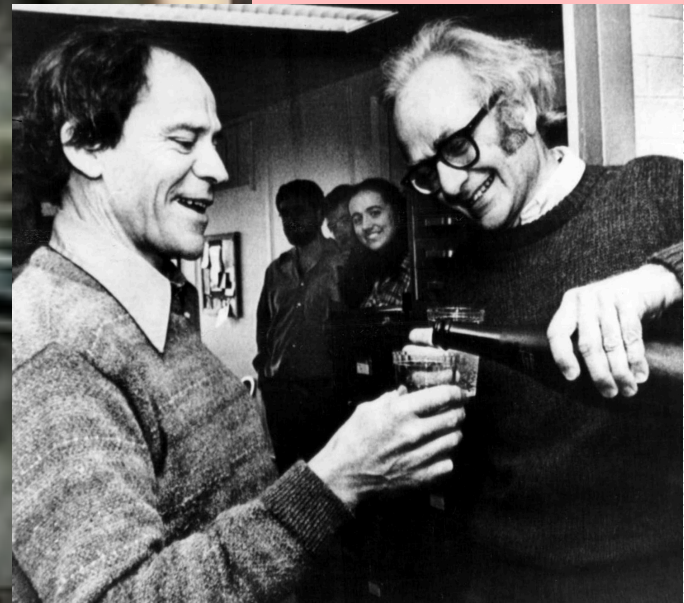


1960年代

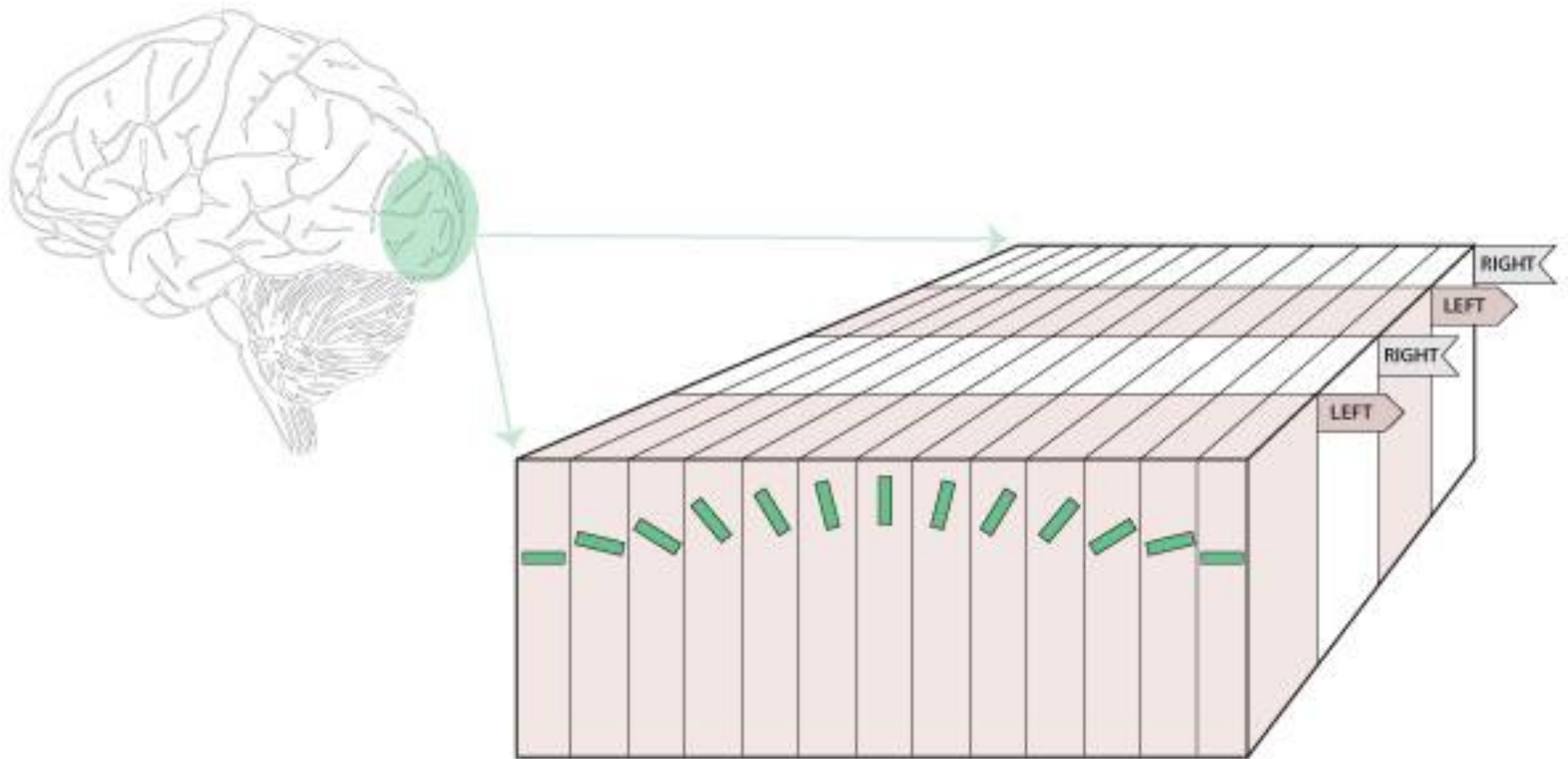
David H. Hubel

Torsten Wiesel

大脳視覚野の研究



60年代から70年代にかけての、Hubel と Wieselの
大脳視覚野の研究は、各方面に大きな影響を与えた。
ニューラル・ネットワークの最初の研究も、こうした影響の
もとで始まった。



Receptive Field of Single Neuron

A bit of history:

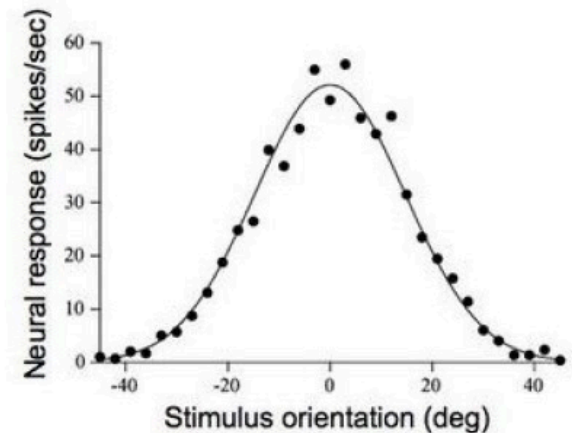
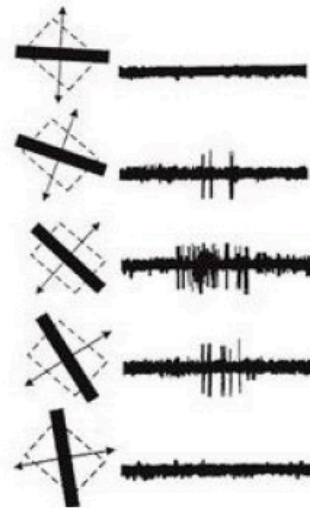
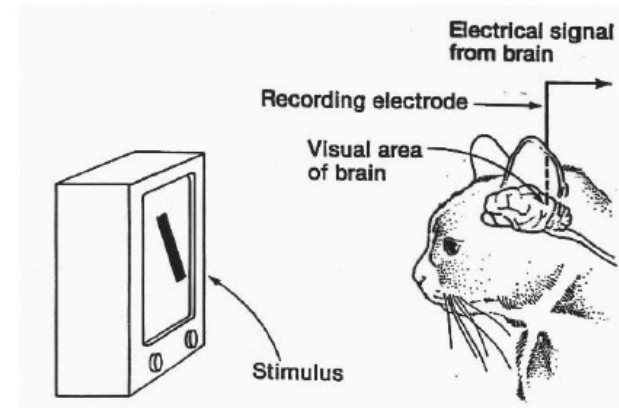
**Hubel & Wiesel,
1959**

RECEPTIVE FIELDS OF SINGLE
NEURONES IN
THE CAT'S STRIATE CORTEX

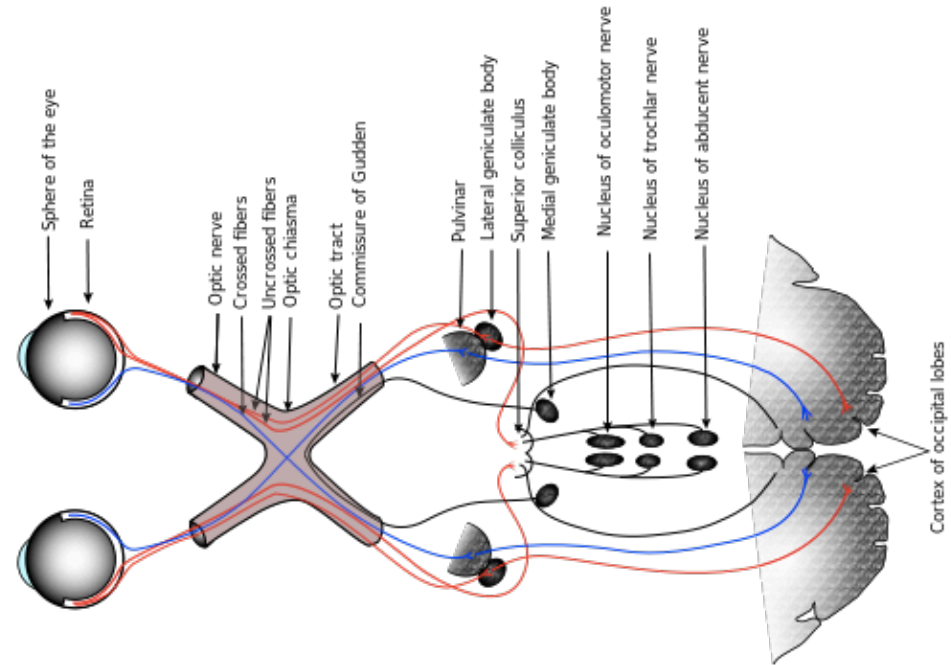
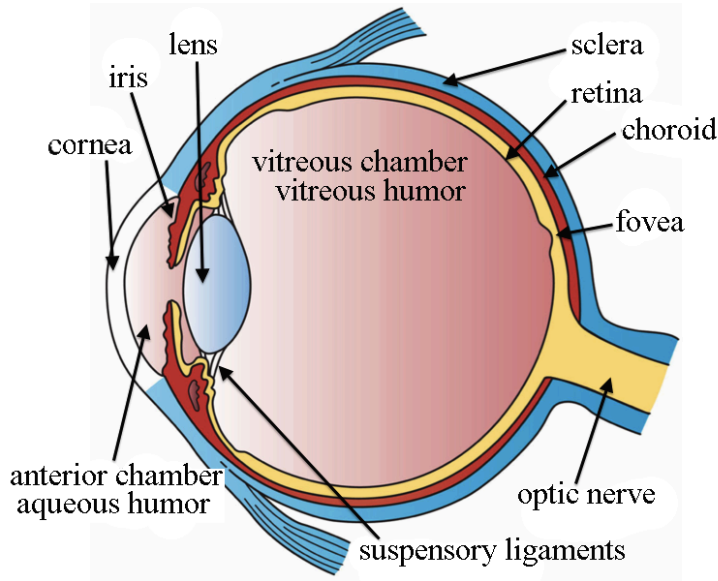
1962

RECEPTIVE FIELDS, BINOCULAR
INTERACTION
AND FUNCTIONAL ARCHITECTURE IN
THE CAT'S VISUAL CORTEX

1968...



視野と大脳視覚野(V1)における、空間情報の正確なマッピング



一次視覚野 (V1)

ヒト成人の一次視覚野の平均ニューロン数はそれぞれの大脳半球につき、約1億4000万個ほどであると見積られる。V1 は視覚における空間情報の明確なマップを持つ。視野と V1 における位置の対応は非常に正確で、盲点でさえも V1 でマップされている。この対応関係は進化的に基本的で、V1 をもつ殆どの動物において見られる。

Hinton, LeCun, Bengioの “Deep Learning”からの抜粋

「ConvNetのコンボリューション層とプーリング層は、視覚神経科学におけるシンプル細胞とコンプレックス細胞に、直接にインスパイアされたものだ。全体のアーキテクチャーは、視覚野の腹側経路のLGN-V1-V2-V4-IT という階層を思い浮かばせるものだ。ConvNetモデルとサルが同じ絵を見せられた時、ConvNetの高位ユニットの活性化は、サルの下側頭皮質の160のニューロンのランダムな集合の変異の半分を説明するものだ。」

<http://goo.gl/HRnXBy>

三次元のデータのかたまりを 三次元のデータのかたまりに 変換するネットワーク

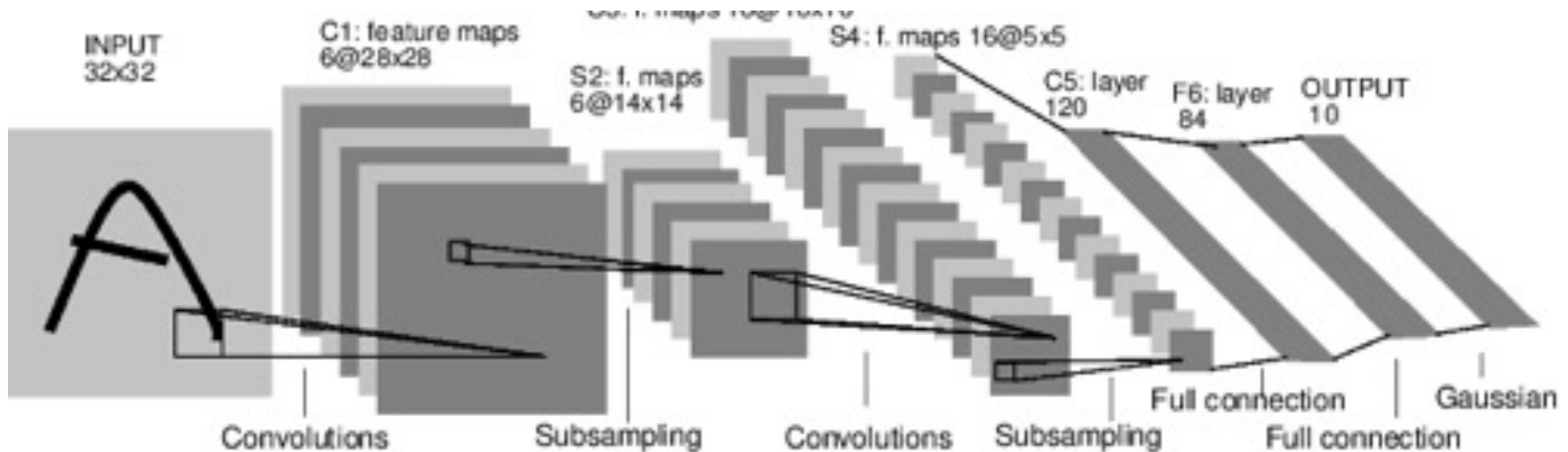
「画像認識専用ニューラルネット」として、ネットワークを、さらにチューンしよう。カラー画像データは、幅×高さ×深さ(RGB)の三次元の構造をしている。CNNは、それぞれの階層で、三次元のデータのかたまりを三次元のデータのかたまりに変換する、ニューラル・ネットワークである。

三次元のボリウムから 三次元のボリウムへの変換

- 基本的には、Full Connect ニューラル・ネットワークのように、無次元の集合から無次元の集合への変換としてではなく、三次元のボリウムから三次元のボリウムへの変換として、ニューラル・ネットワークを再構築する。
 - もちろん、最初の入力層に置かれるのは、幅 × 高さ × 深さ(RGB)の三次元の座標位置を持つ、生のカラー画像である。
 - CNNは、入力をカラー画像と想定して、認識専用ニューラルネットとしてチューンされたニューラル・ネットワークである。
-

三次元のボリュームから三次元のボリュームへ

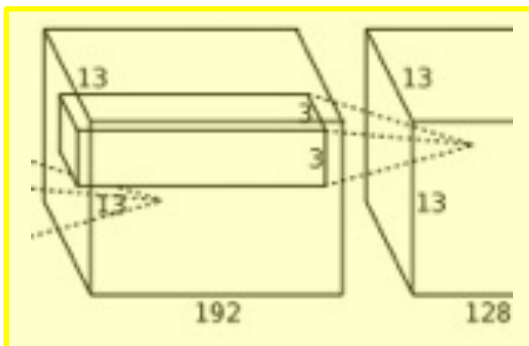
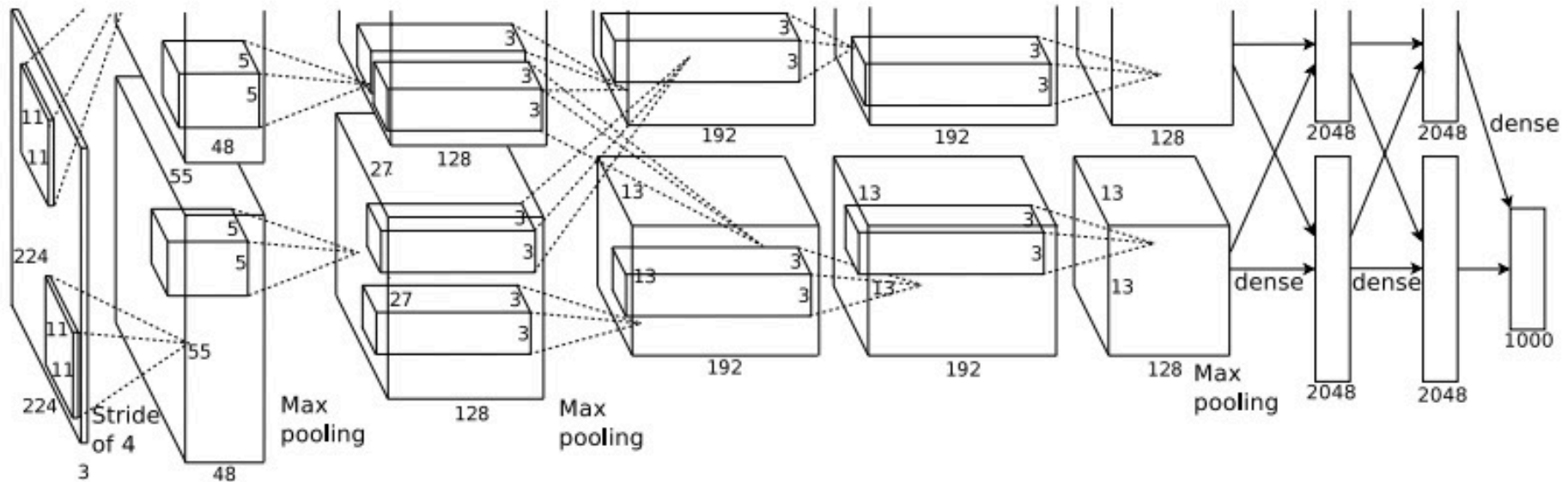
LeNet



平面が、複数枚、重ねて置かれているが
四角錐は、何を表しているのか？

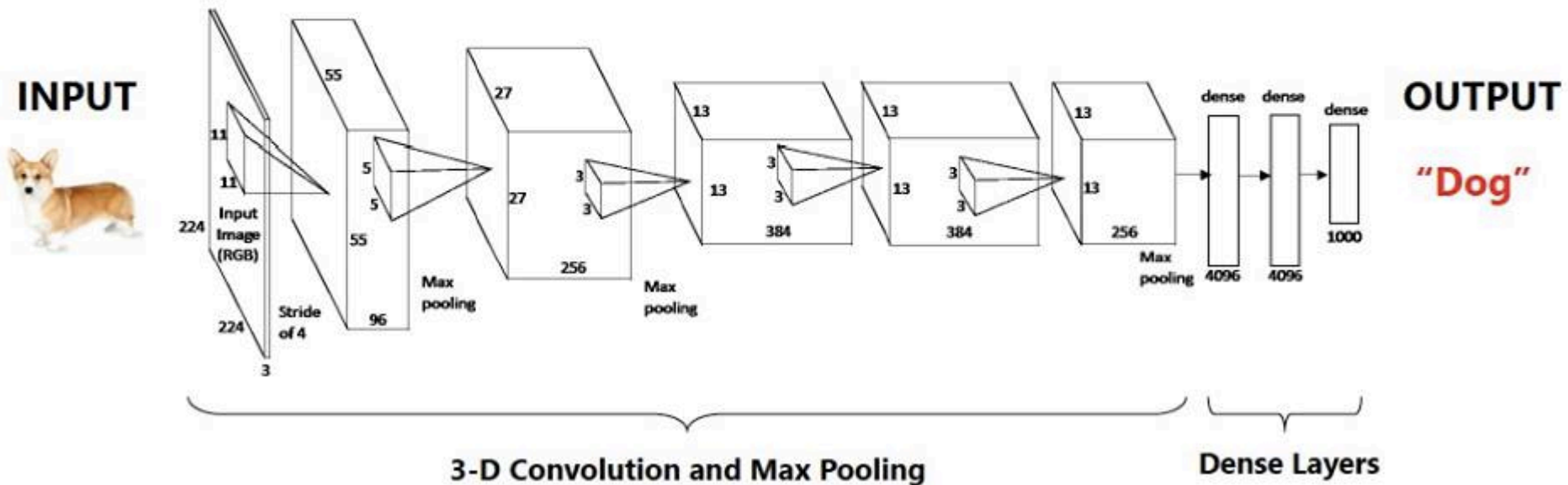
三次元のボリュームから三次元のボリュームへ

AlexNet



ここでは、もう、全部、立体の直方体になっている。
小さな直方体の上に、四角錐がおかれている。
この小さな直方体は、何を意味しているのだろうか？

三次元のボリュームから三次元のボリュームへ



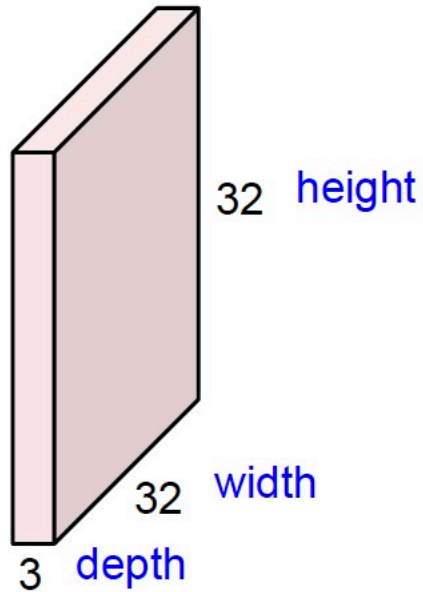
これも、全部、立体の直方体になっている。
小さな直方体は、省略されている。
四角錐の位置が、先のととは異なっている。

CNNの三次元ボリューム 構成の概要を見る

先の図に現れた直方体や四角錐は、何を表している
のだろうか？ここでは、それを見てみよう。

Convolution Layer

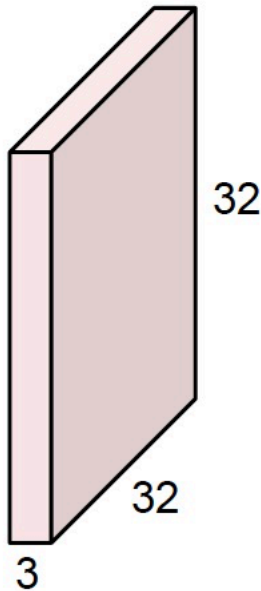
32x32x3 image



入力データは、32x32のカラー画像だとして
しょう。入力層は、32x32x3の三次元の
ボリュームからなる。

Convolution Layer

32x32x3 image



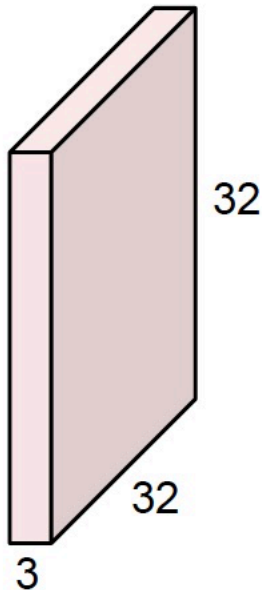
5x5x3 filter



フィルターは、5x5x3のボリュームを持つとしよう。フィルターは、画像上を移動しながら、対応する要素ごとに内積を計算してゆく。

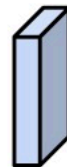
Convolution Layer

32x32x3 image



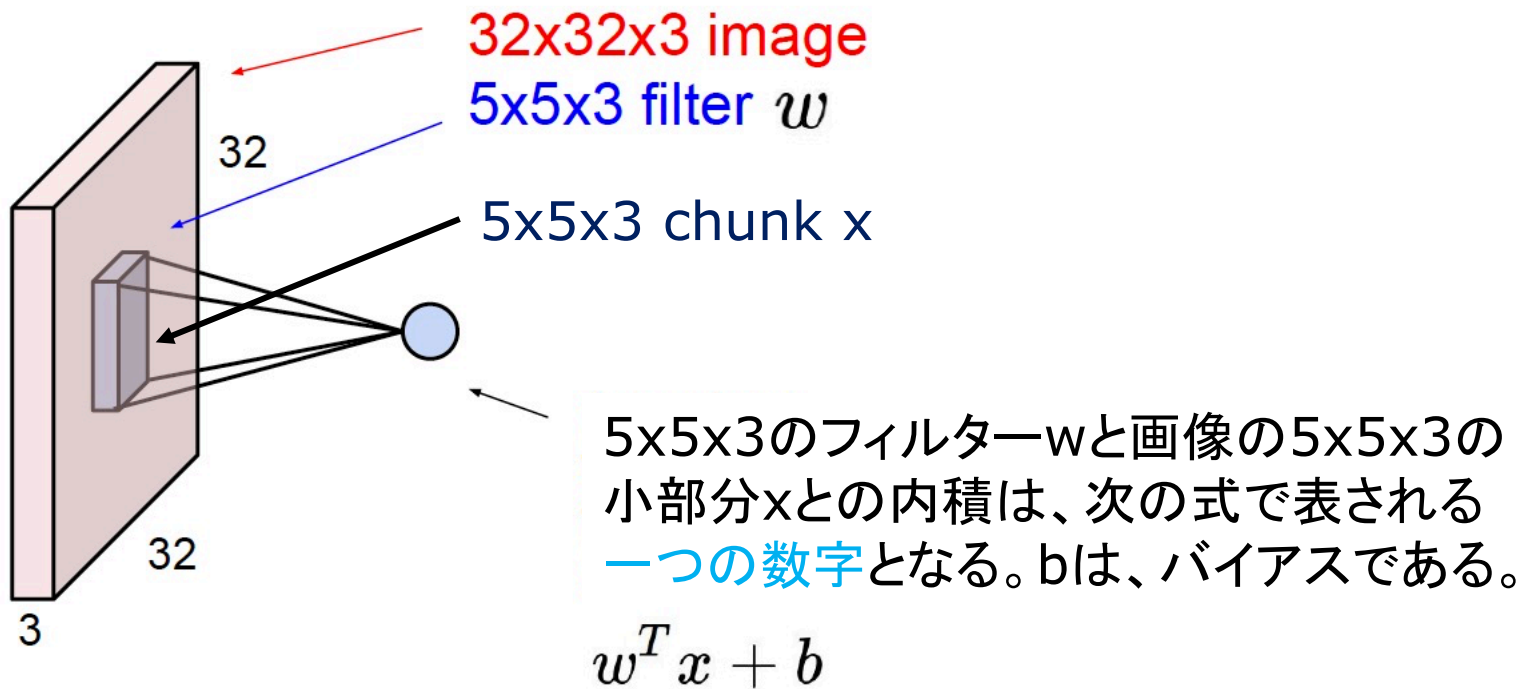
フィルターは、常に、入力ボリュームと同じ「深さ」を持つ。ここでは、3である。

5x5x3 filter



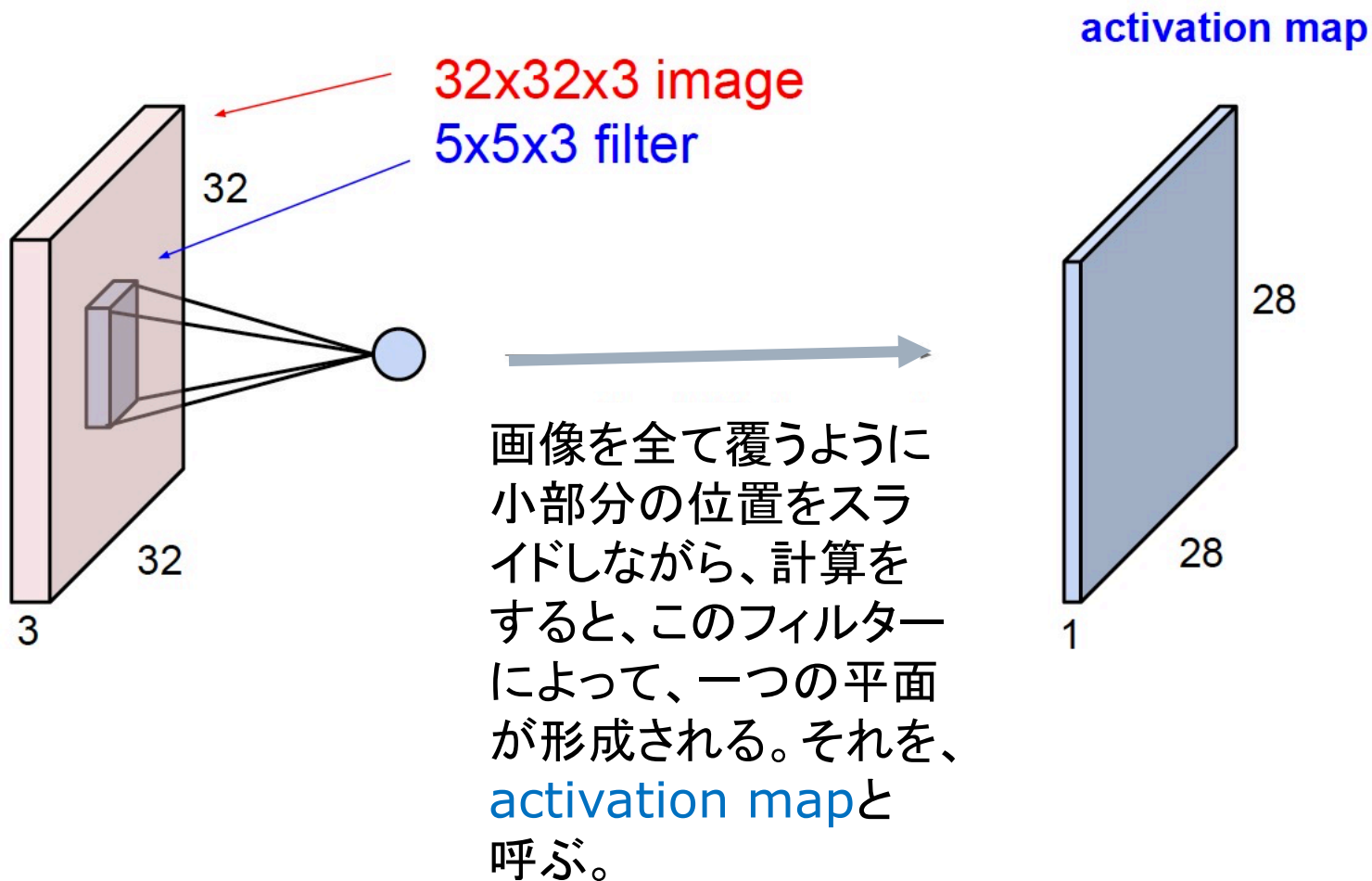
フィルターは、5x5x3のボリュームを持つとしよう。フィルターは、画像上を移動しながら、対応する要素ごとに内積を計算してゆく。

Convolution Layer



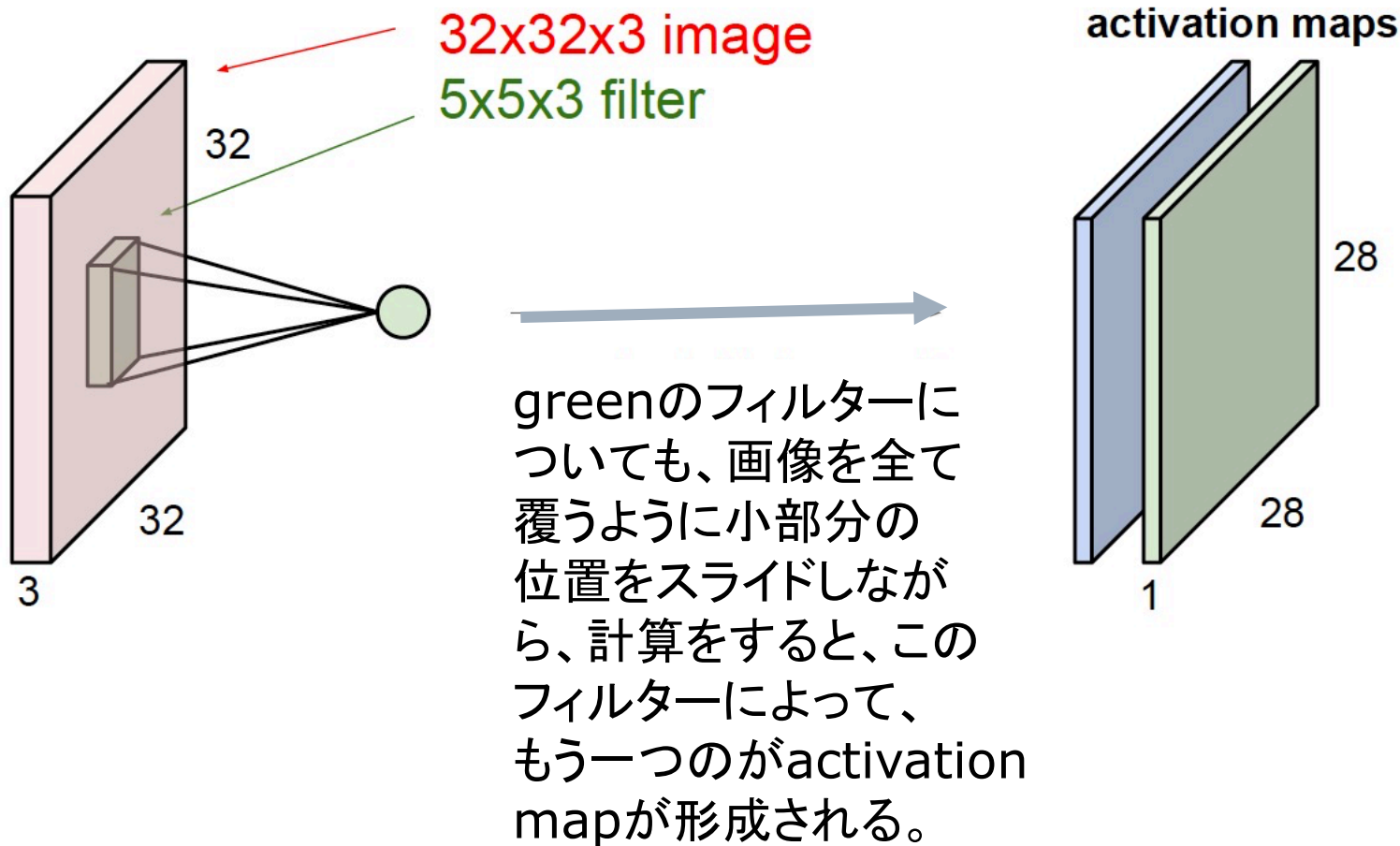
画像上の小部分の位置に対応して、
一つの数字が計算される。

Convolution Layer



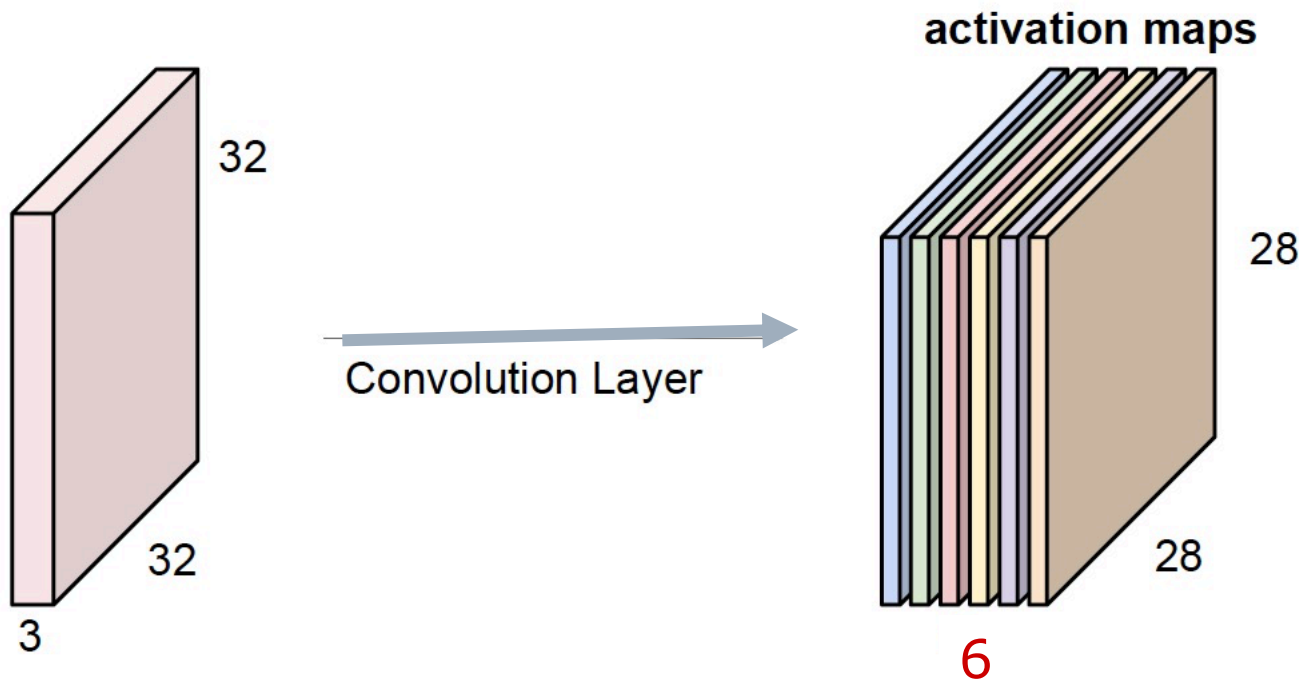
Convolution Layer

二つ目のgreenのフィルターを考える



Convolution Layer

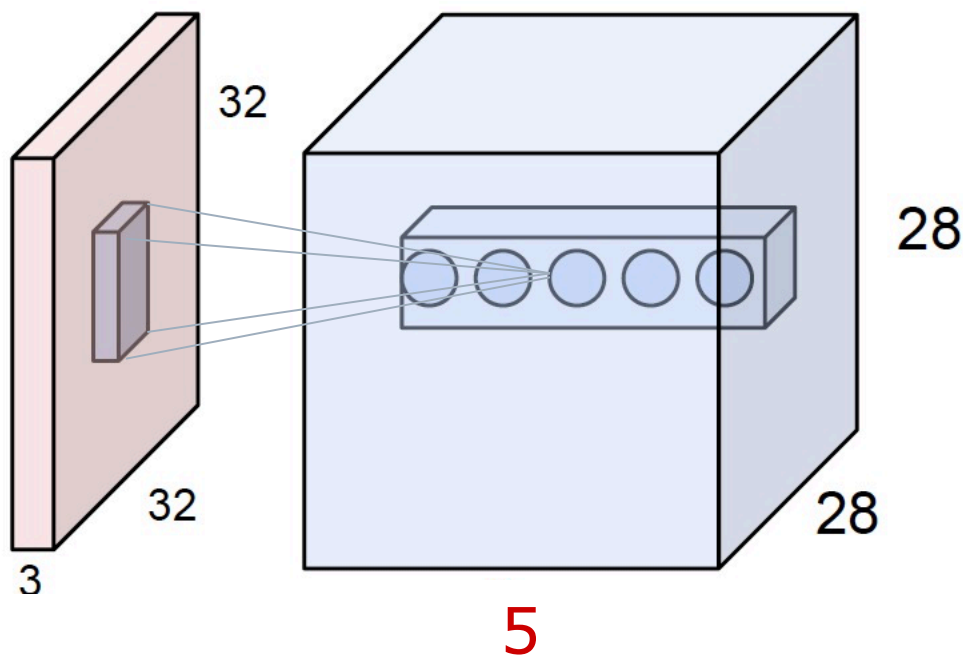
もし、**6**個の $5 \times 5 \times 3$ のフィルターを、画像に適用すれば、別々の**6**個のactivation mapを得ることになる。



これらを積み重ねて、 $28 \times 28 \times 6$ のサイズの「新しい画像」を得る。Convolutionの結果得られる画像の「深さ」は、適用したフィルターの個数に等しくなる。

Convolution Layer

もし、**5**個の $5 \times 5 \times 3$ のフィルターを、画像に適用すれば、
別々の**5**個のactivation mapを得ることになる。

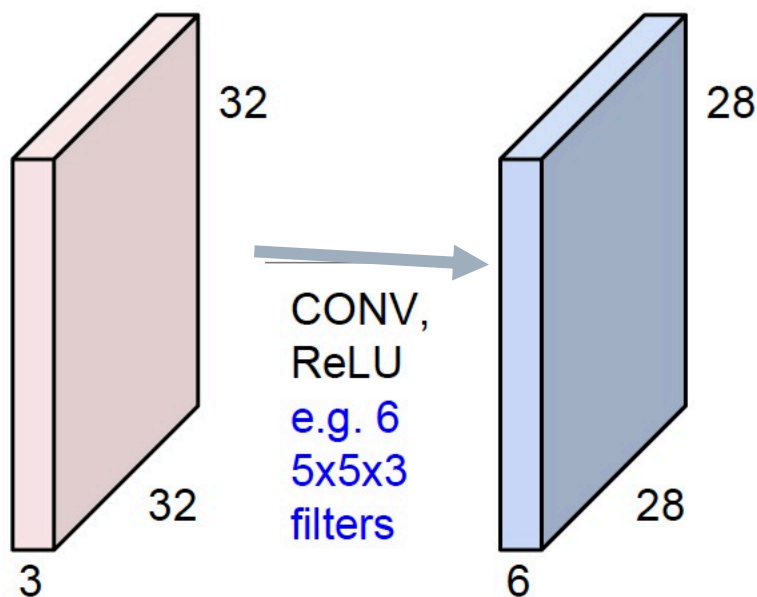


5個のフィルターでは、
CONV層は、 $28 \times 28 \times$ **5**
に、格子状に配置された
ニューロンから構成され
ることになる。

5つの異なったニューロン
が、入力層の同じ $5 \times 5 \times 3$
の小さな領域のデータを見ている。

Convolution Layer

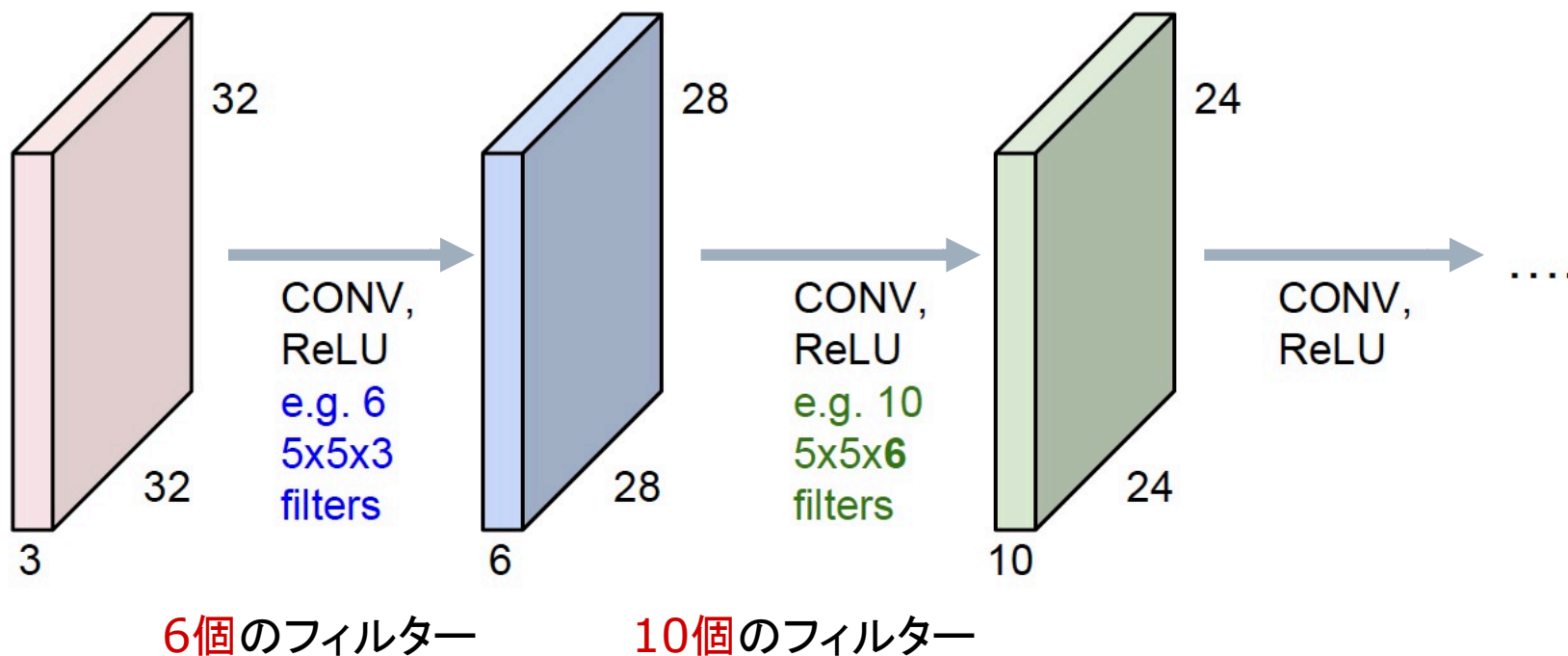
Convolutional Neural Networkは、あいだにReLUのような活性化関数を置いた、Convolution層の連なりである。



6個のフィルター

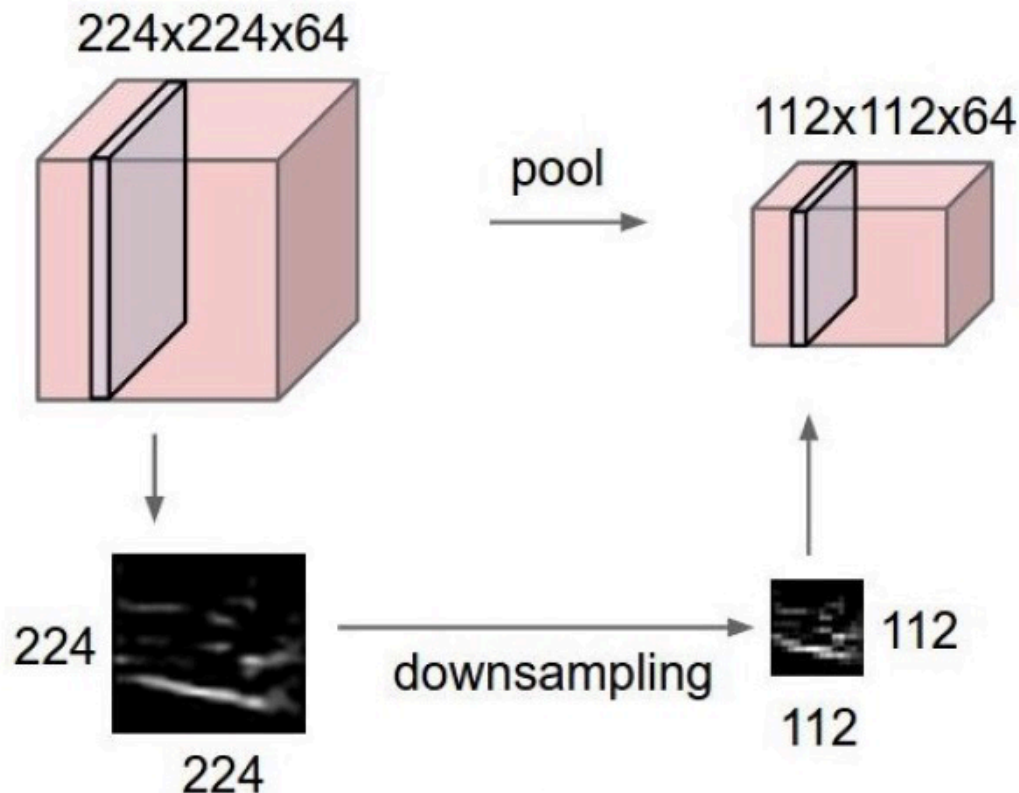
Convolution Layer

Convolutional Neural Networkは、あいだにReLUのような活性化関数を置いた、Convolution層の連なりである。



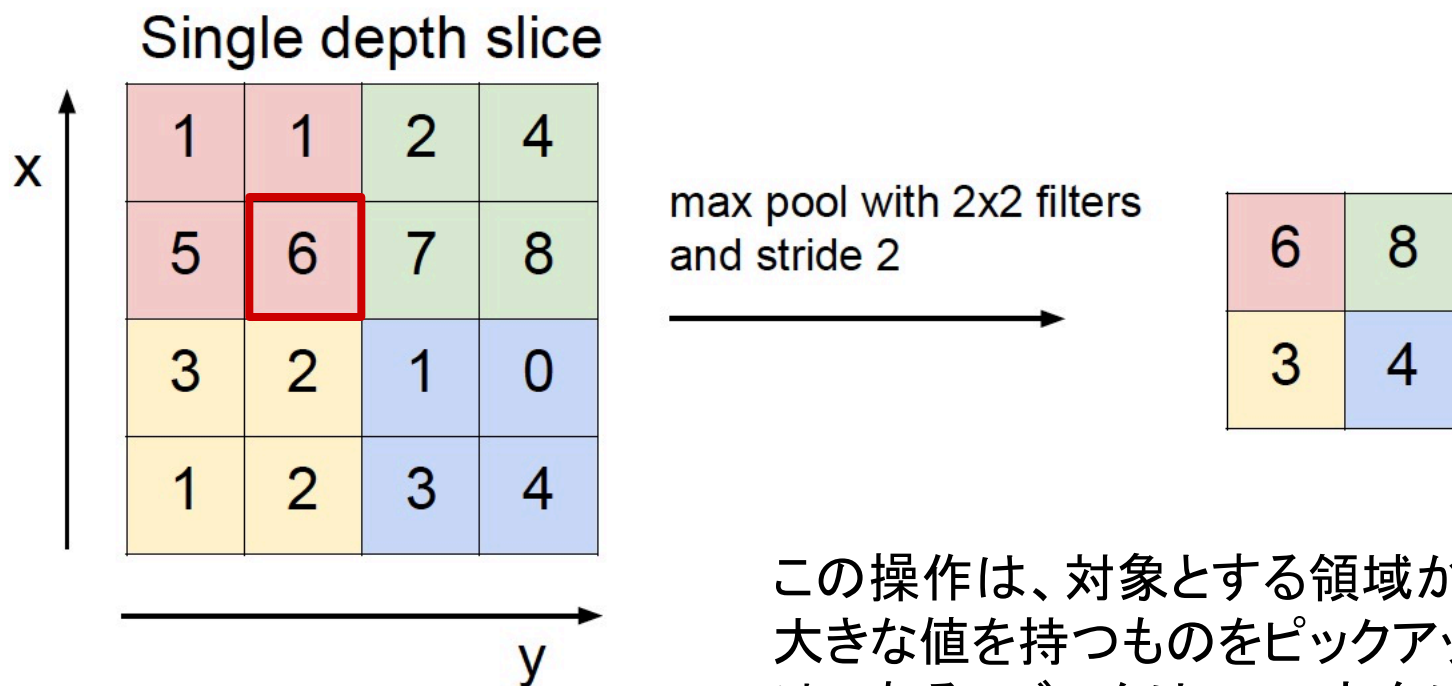
Pooling layer

Pooling層は、表現をもっと小さくして、扱いやすいものにする。
それぞれのActivation Mapに独立に作用する。データの微細なシフトや欠損は、捨象される。



Pooling layer

MAX POOLING



この操作は、対象とする領域から、一番大きな値を持つものをピックアップするだけである。データは、x-y方向に関しては、縮小される。（「深さ」は、変化しない。）
この操作は、パラメーターを持たない。