

A person with long dark hair, wearing a white long-sleeved shirt and a black skirt, stands on a sandy dune. They are looking out at a vast blue sky with scattered white clouds. The person is carrying a light-colored bag. The scene is captured from behind, showing their silhouette against the bright sky.

ディープラーニング入門 集中講義

丸山不二夫

本集中講座の構成

Part I

ニューラルネットワークの基礎

Part II

ニューラルネットワークは、
どう「学習」するのか？

Part III

ニューラルネットワークによる
画像認識技術



資料URL

Part I <https://goo.gl/SLF7dB>

Part II <https://goo.gl/YWk5rj>

Part III <https://goo.gl/Mrd49F>



ディープラーニング入門集中講義

Part I

ニューラルネットワークの基礎





まず、学ぶことから始めよう！

ITの世界は、この10年で大きく変わった クラウドとモバイルの時代

- 2004年 Google上場
- 2006年 Amazon EC2, S3
- 2007年 Apple iPhone
- 2008年 Microsoft Azure
- 2008年 Google Android
- 2012年 Facebook上場

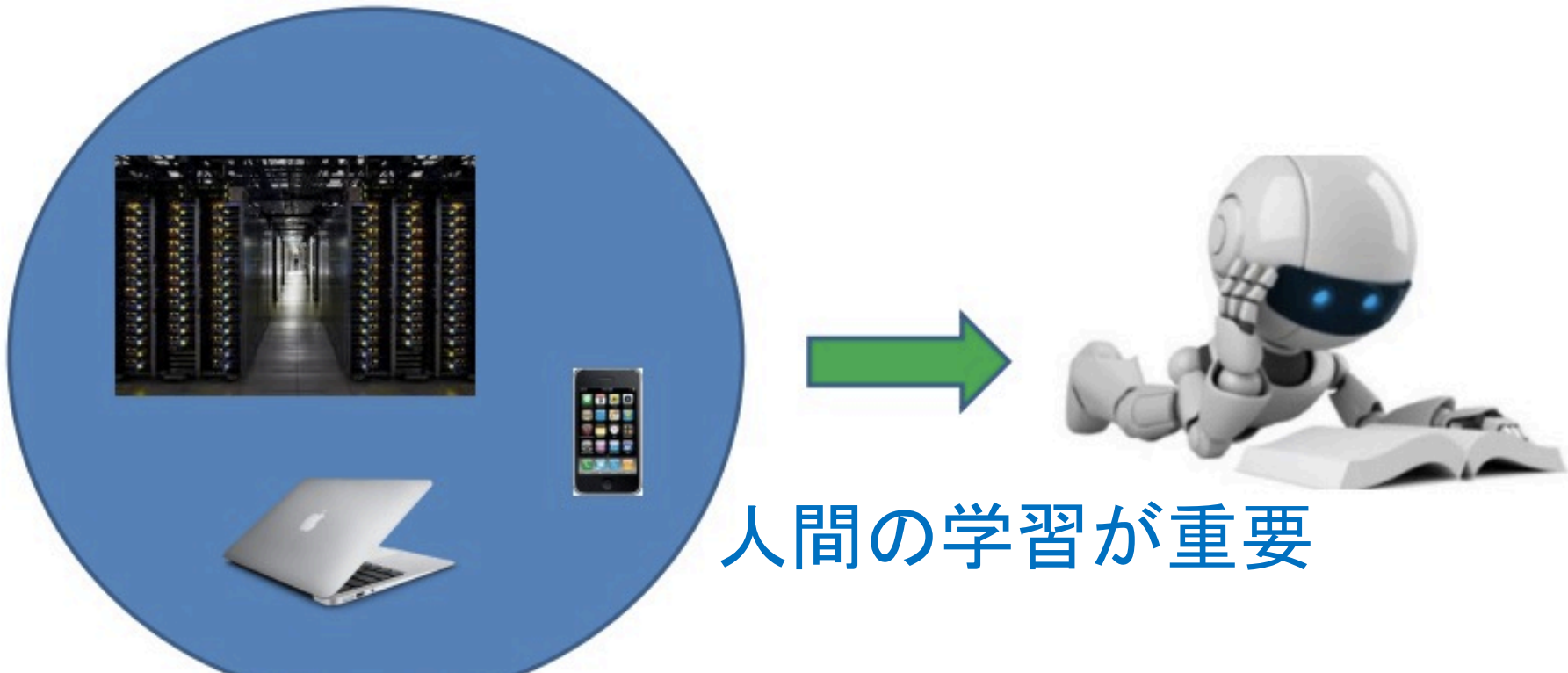
世界は、この先10年でどう変わるのか？

来たるべきAI新時代に向けて

- モバイルとクラウドの時代が始まって10年が経った。次の10年を展望した時、AI技術が、その新しい技術的中核として登場しようとしている。
 - ただ、モバイルが携帯電話の、クラウドがデータセンターの、ある意味では連続的な発展だったのに対して、ニューラル・ネットワークは、ほとんどの技術者にとって全く新しい技術である。そこには、技術的な「断絶」がある。
 - 問題は明確である。我々は、来たるべきAI新時代に向けて、学ぶことから始めなければならない。同時に、重要なことは、今後の10年を展望すれば、それはすべての技術者にとって必要な課題になるだろうだということである。
-

Paradigm Shift: From Computers to Learning Machines

We are in the very beginning of new computer era



CAMEL の Opening Remark のスライド <http://goo.gl/JVjGne>

まず、学ぶことから始めよう。
人工知能を作るには、人間の知能
が必要であり、機械学習を知るため
には、人間の学習が必要である。



様々なAI技術

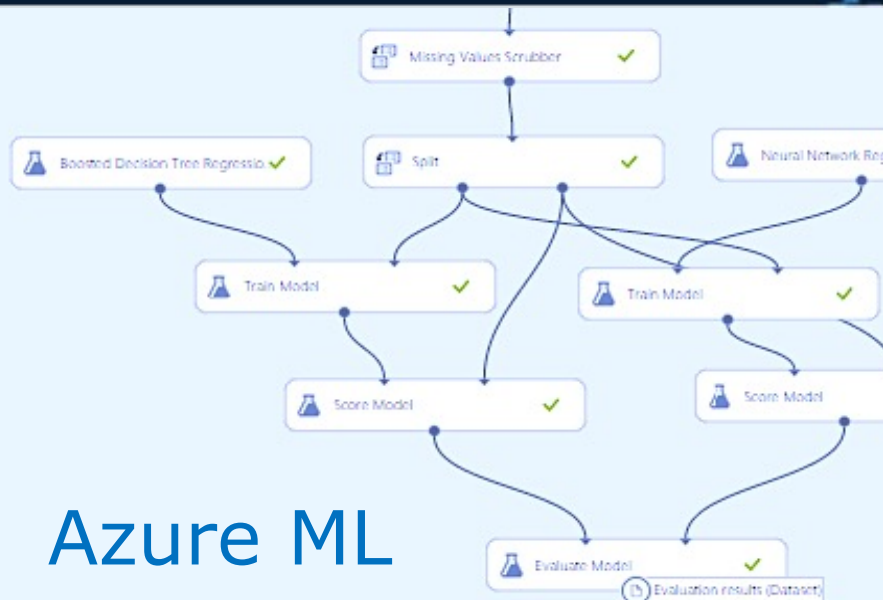
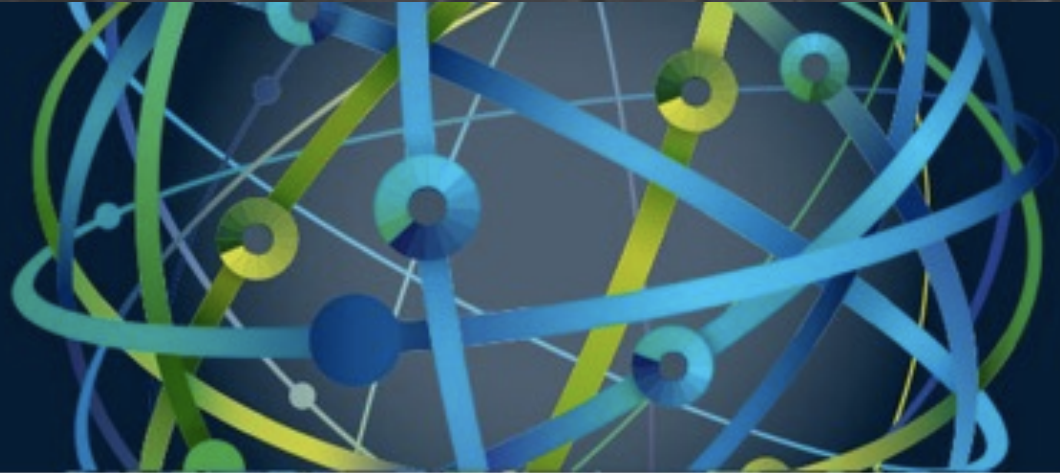


Prediction API

使いやすい RESTful インターフェースを通じて Google の機械学習アルゴリズムを利用できる、データの分析と予測のための API です。

Announcing IBM Watson Analytics

Learn how Watson Analytics delivers powerful analytics to every business user



Azure ML

統計処理



A. 統計的分析に基づく、数値予測・クラス分け

- これは昔から統計学の手法として存在する、回帰分析やクラスター分析と考え方は、基本的に同じものである。
 - ただ、マシンのパワーが飛躍的に増大して、大量のデータを短時間で処理できるようになり、かつ効果的なビジュアライゼーションが可能になったのが、実践的には新しく、かつ重要な変化である。
 - 現在、クラウドで提供されている、大部分の機械学習のサービスは、このクラスに属する。
 - 同時に、「決定論的なアプローチ」に対する「統計的なアプローチ」の優位という方法論的な自覚は、現在の機械学習技術・AI技術の多くの領域に、大きな影響を与えつつある。
-



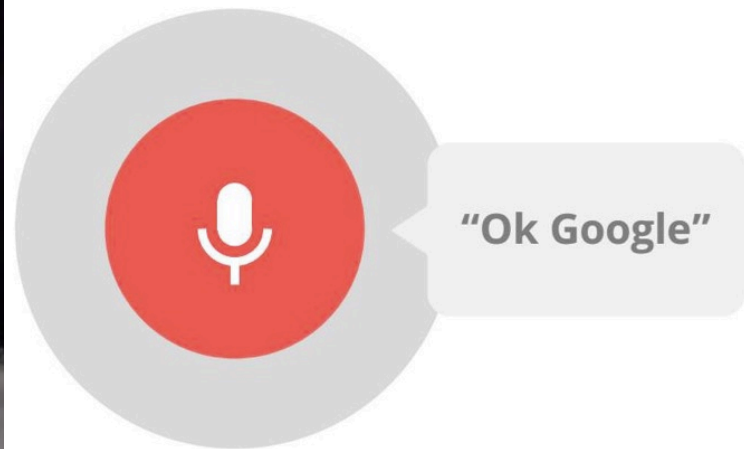
ドイツの侵攻を受け
ロンドンへ亡命したオランダの女王は？

パーソナル・アシスタンス・システム

Amazon Alexa



Microsoft Cortana



B. 経験的に構築された知識・推論・対話システム

□ パーソナル・アシスタント・システム

GoogleのGoogle Now, AppleのSiri, MicrosoftのCortana, AmazonのAlexa等々。私たちの身近なところで、もっとも活発に応用が進んでいる分野。完成度の高い「質問応答システム」として、(かつての)IBMのWatsonがある。

□ 入力(Speech to Text)、出力(Text to Speech)に、ニューラル・ネットワーク技術を利用する。

□ 「パーソナルアシスタント」のプログラムは、Turing Test的には、「知的」にふるまっているように見えるのだが、その多くの実装は、「知的」なものとは言えない。

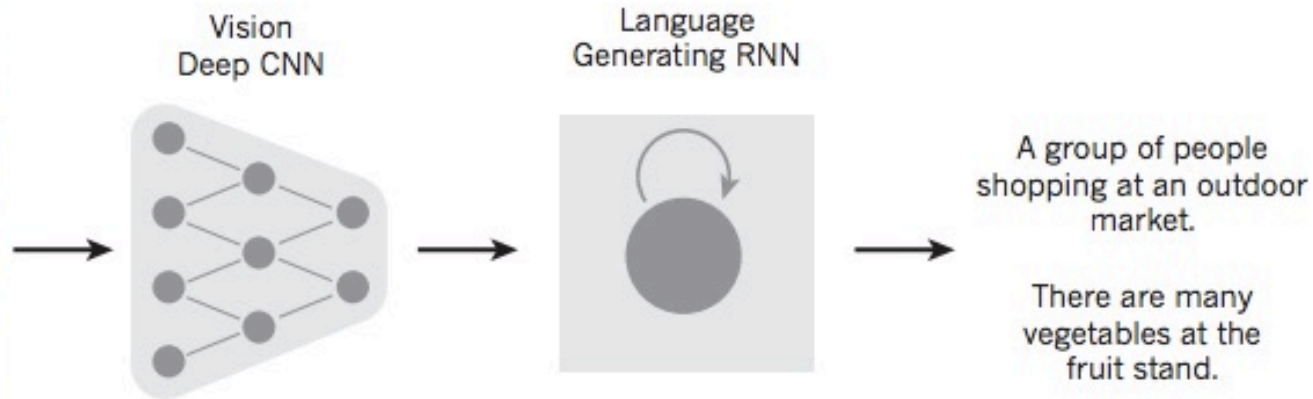


Deep Learning

C. ニューラルネットワークに基づく、 知覚・運動系へのアプローチ。Deep Learning

- 現在、活発に研究・応用が進んでいる分野である。文字認識・発話の文字列への変換、静止画・動画からの対象の判別・動きの検出等々で、大きな成功を収めている
 - 基本的なモデルの起源は、生物の知覚・感覚（主要には、視覚・聴覚）を司る神経回路網にある。
 - 生物の知覚・感覚の能力は、運動能力と密接に結びついて、感覚・運動系を構成しているので、ロボットの動作の制御、ドローンの姿勢制御、自動運転等にも、このアプローチは有効で、応用分野を拡大している。
-

イメージからテキストへ

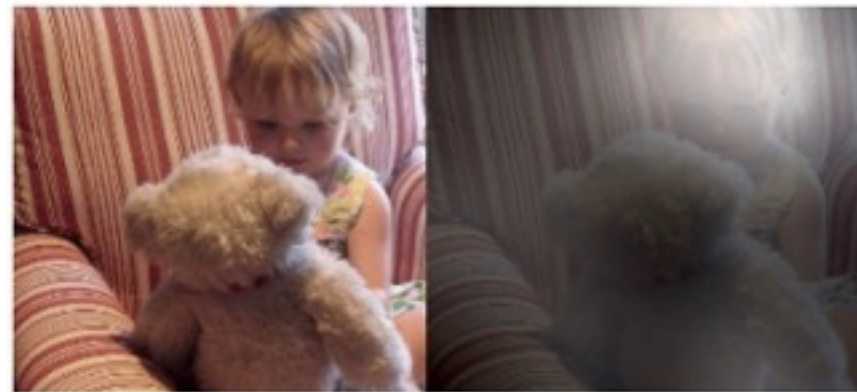


「一群の人たちが野外マーケットでショッピングをしている」
「フルーツスタンドにはたくさんの野菜がある」



A woman is throwing a **frisbee** in a park.

「女性が公園で**フリスビー**を投げている」



A little **girl** sitting on a bed with a teddy bear.

「**幼**い子がテディベアを持ってベッドに座っている」

言語理解

D. 人間固有の言語能力の機械による実現

- 「機械翻訳システム」 本来は、意味理解を含む人間の一般的な言語能力に関わっている。ただ、機械翻訳の場合、質問に答えることが求められているわけではない。それは、ある言語から他の言語への形式的な「書き換え」システムなのかもしれない。
- この領域は、最終的には、機械が人間のように、言語の「意味」を理解して、「知識」を活用して「意味」のある活動をするのを目指している。ある意味、人工知能の本丸である。ただ、多くの課題が残されている。

HoTT(Homotopy Type Theory)と Univalent Theory

- 論理＝数学的には、 $a : A$ には、様々な解釈がある。ここでは、他の解釈と比較して、HoTTでの $a : A$ の解釈を見てみよう。

数学的認識の理解

- 
1. 集合論: Russellの立場
Aは集合であり、aはその要素である。
 2. 構成主義: Kolmogorovの立場
Aは問題であり、aはその解決である
 3. PAT: Curry & Howardの立場
Aは命題であり、aはその証明である。
 4. HoTT: Voevodskyの立場
Aは空間であり、aはその点である。

"Propositions as Types"
"Proofs as Terms"

E. 人間の数学的・論理的な推論能力に関わるもの

- ここでは、経験的な起源からは相対的に独立な、純粹に数学的な定理の証明や、複雑な証明の機械による支援が課題になる。この分野でも、COQ等の「証明支援システム」を中心に、興味ふかい前進がある。
 - マルレク「型の理論」と証明支援システム -- Coqの世界」
<http://bit.ly/1qbdXVE> を参照されたい。
- かつての日本の「第五世代」的な、AIに対する「論理的」な(というより「定理証明的」な)アプローチは、現在のAI研究においては主流ではない。ただ、そのことは、他のすべての領域と同じように、数学的・論理的なアプローチの有効性を否定するものではない。
- 数学や物理といった知識の体系は、誰もが生得的に持つ言語能力や経験的な知識の体系には、還元されない。

AI技術を構成する複数の流れの中で、なぜ、ニューラル・ネットワークが重要なのか？

- A) 既知のデータの統計的分析をもとに、数値予測・クラス分けを行うもの。**機械学習技術**。
- B) 経験的に構築された知識・推論・対話システム。**パーソナル・アシスタンス・システム** etc。
- C) ニューラルネットワークの手法を用いて、生物の感覚・運動系の能力の相同物を機械上で実現しようとするもの。**ディープ・ラーニング**。
- D) 人間固有の言語能力の機械による実現。
- E) 人間の数学的・論理的な推論能力に関わるもの。

ニューラル・ネットワークは、 新しいAI技術の**現在**の技術的中核だから

A) 既知のデータの統計的分析をもとに、数値予測・クラス分けを行うもの。機械学習技術。

NNは、同等の能力を持つ

B) 経験的に構築された知識・推論・対話システム。パーソナル・アシスタンス・システム etc。

Speech2Text
Text2Speech
Human Interface

C) ニューラルネットワークの手法を用いて、生物の感覚・運動系の能力の相同物を機械上で実現しようとするもの。ディープ・ラーニング。

CNN+RNN
機械翻訳
Knowledge Database

D) 人間固有の言語能力の機械による実現。

E) 人間の数学的・論理的な推論能力に関わるもの。

人工知能の未来

- A) 既知のデータの統計的分析をもとに、数値予測・クラス分けを行うもの。機械学習技術。
- B) 経験的に構築された知識・推論・対話システム。パーソナル・アシスタンス・システム etc。
- C) ニューラルネットワークの手法を用いて、生物の感覚・運動系の能力の相同物を機械上で実現しようとするもの。ディープ・ラーニング。
- D) 人間固有の言語能力の機械による実現。
- E) 人間の数学的・論理的な推論能力に関わるもの

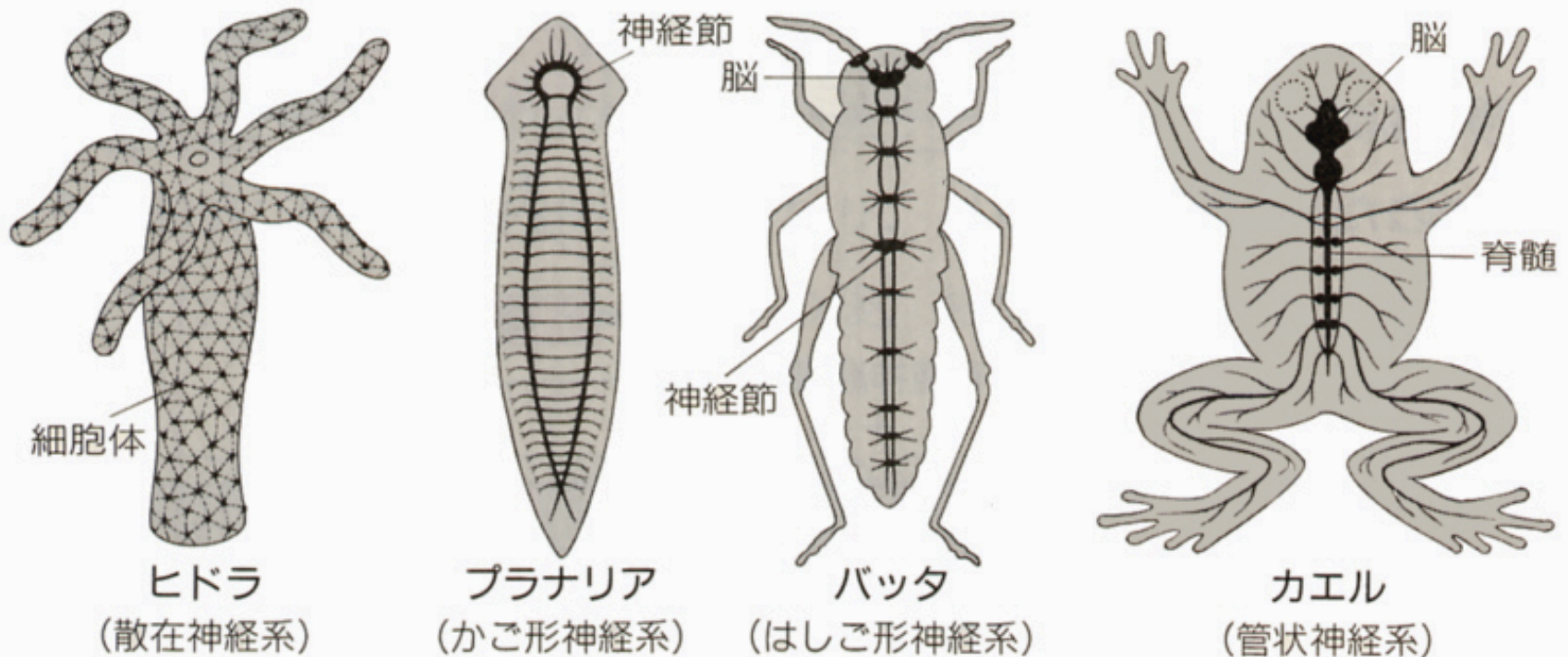


未来の、
「人工知能」



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.

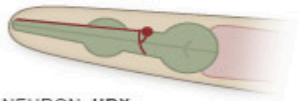
RMG combines information from several sensory neurons, also highlighted, and then signals the worm's muscles to move toward nearby worms if conditions are right.

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

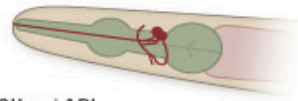
1987年

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

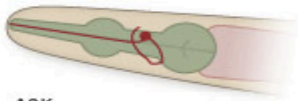
FOLLOWING NEURONS
Six neurons highlighted in the diagram are shown below. Most neurons are located in the head, though some, like AVA, run the length of the worm's body.



NEURON: URX
Senses oxygen in the environment



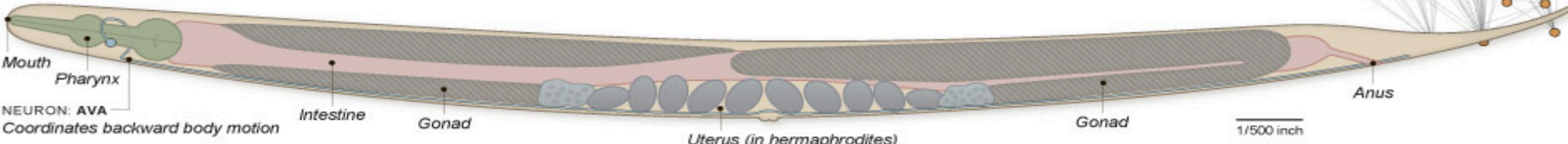
ASH and ADL
Sense poisons and other noxious stimuli



ASK
Senses food signals and detects sex pheromones from other worms



RMG
Hub of a network of neurons that controls social behavior



NEURON: AVA
Coordinates backward body motion

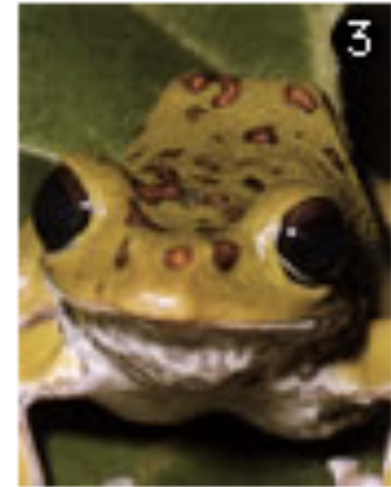
アノマロカリス



「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

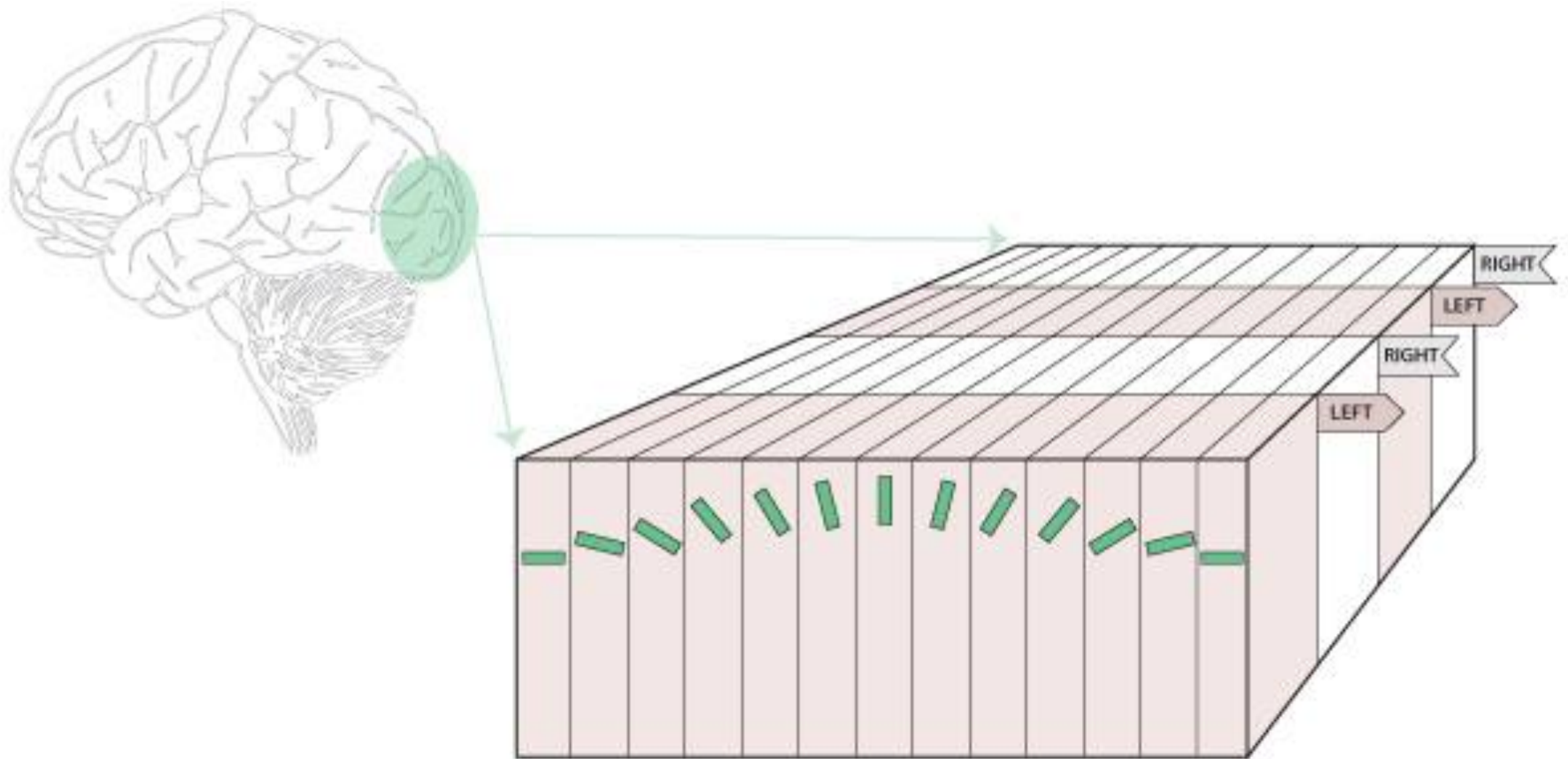


視覚と運動能力は、強く結びついている



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。

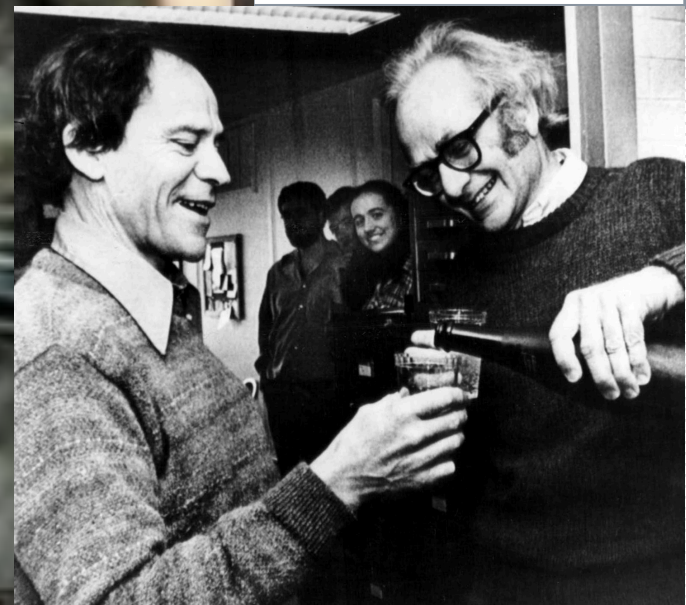




1960年代

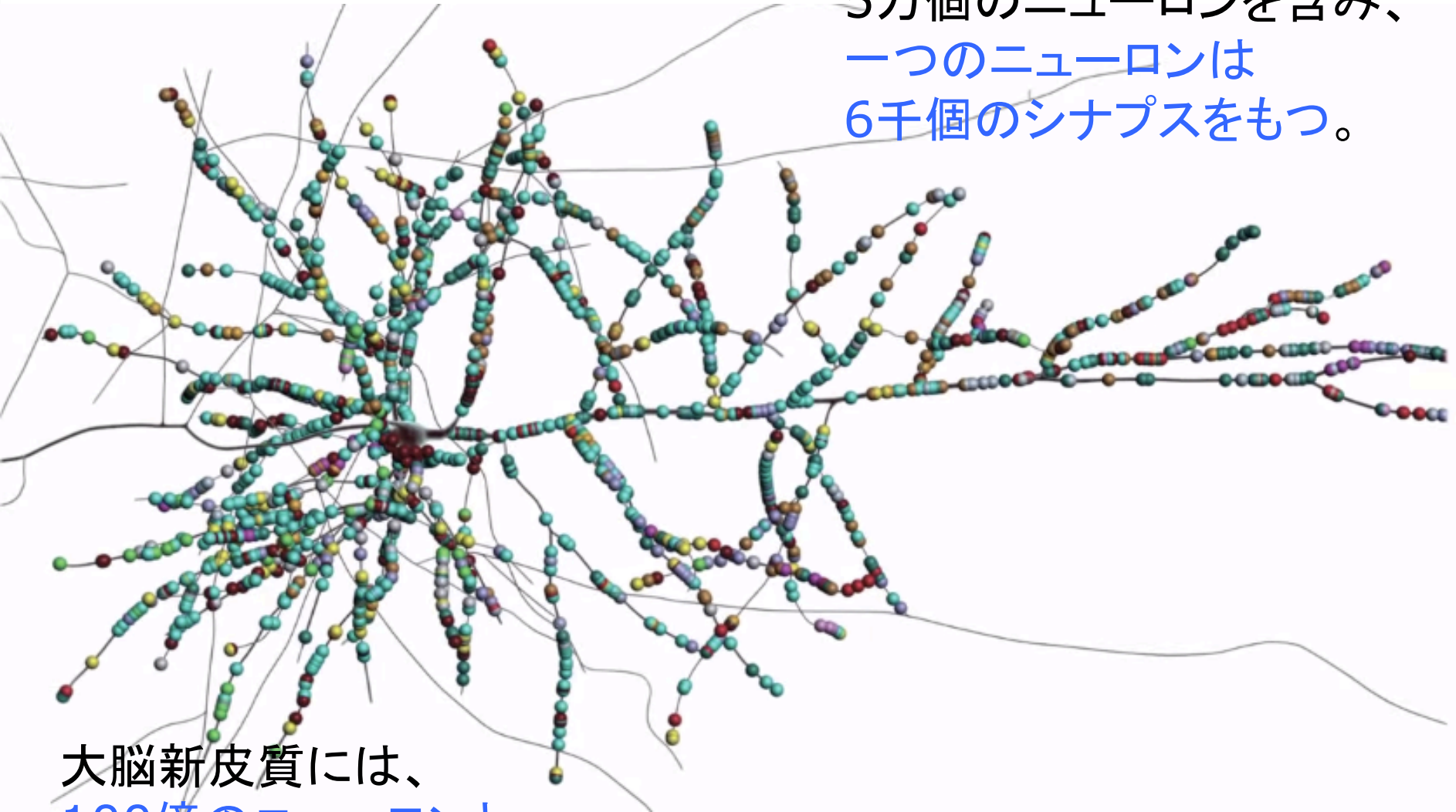
David H. Hubel

Torsten Wiesel





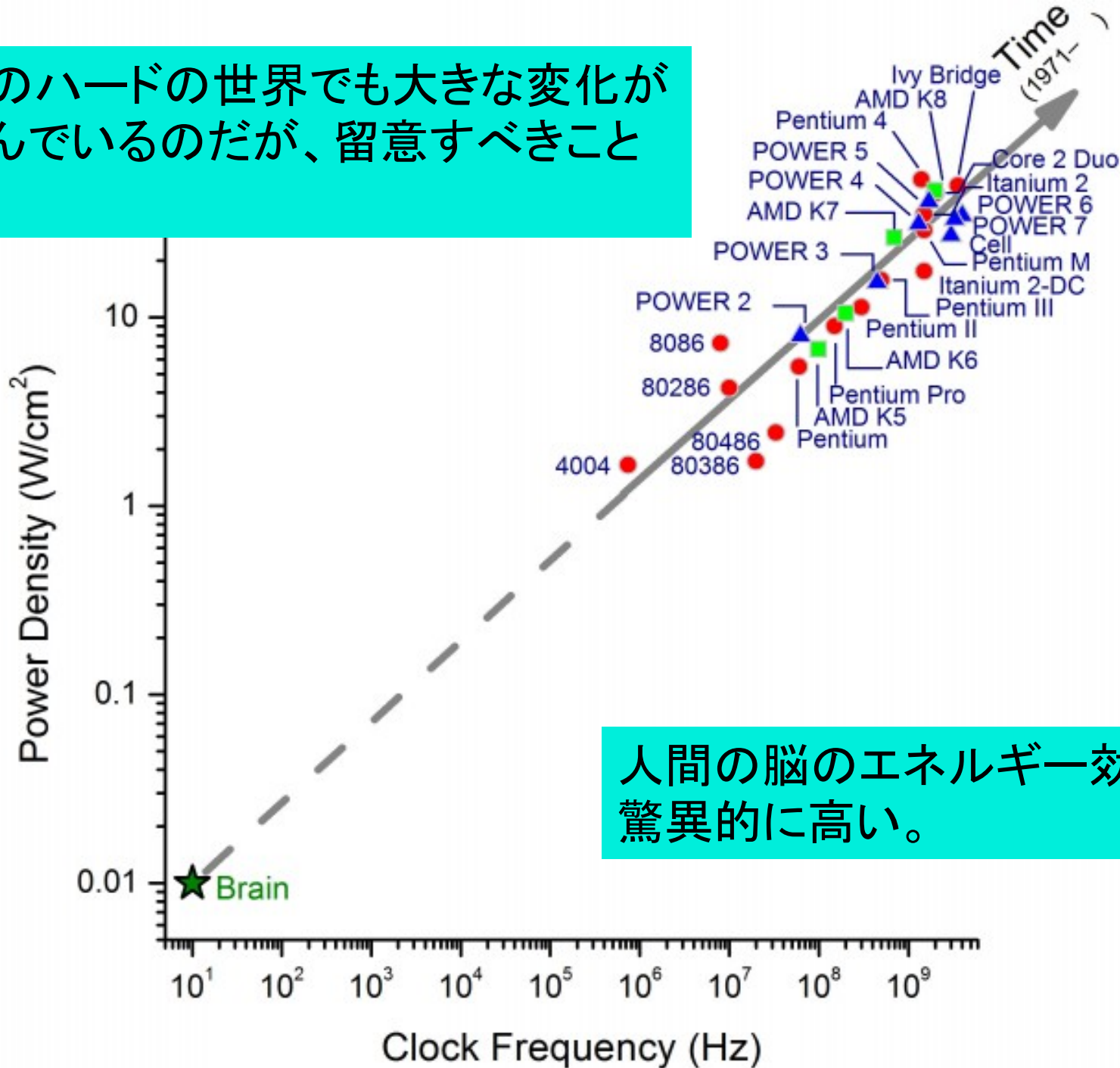
脳は、1mm立方に
5万個のニューロンを含み、
一つのニューロンは
6千個のシナプスをもつ。



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。





脳研究の取り組み

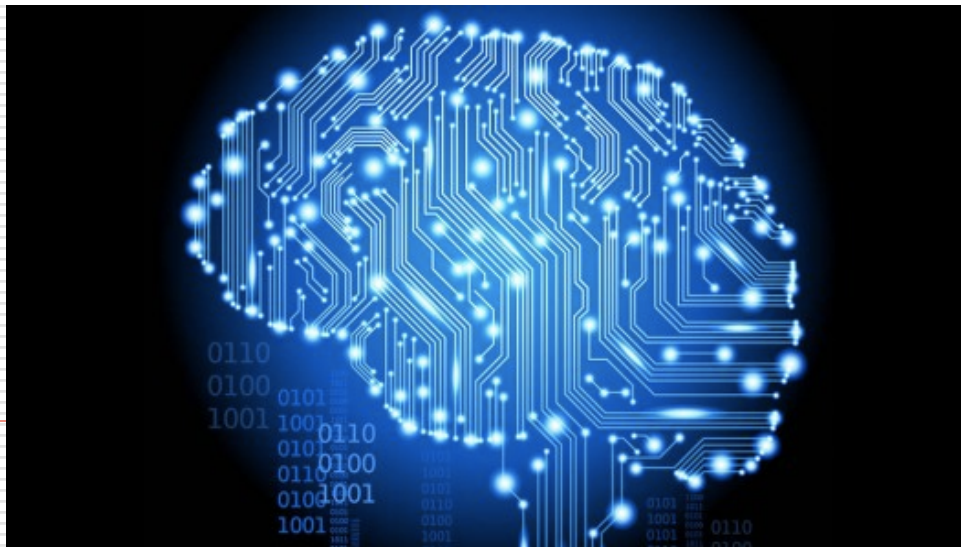
脳研究の取り組み

現在は、それぞれ異なった道を歩いているのだが、ニューラル・ネットワークの研究と脳研究とは、「人間の知能」の解明という共通の目標で結ばれている。ここでは、脳研究のいくつかの取り組みを紹介する。

Blue Brain Project

Henry Markram

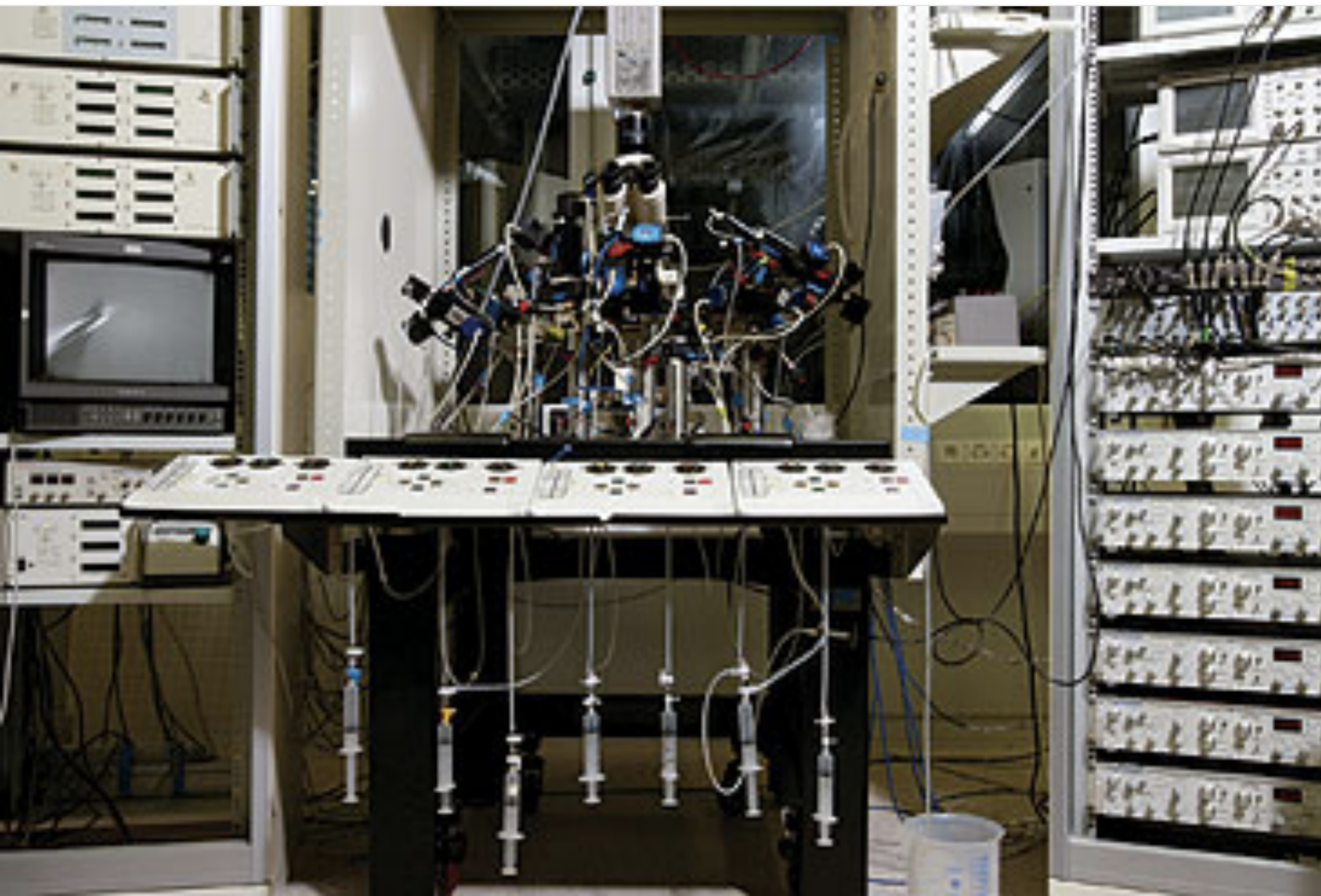
- "I wanted to **model the brain** because we didn't understand it."
- "The best way to figure out how something works is to **try to build it from scratch.**"



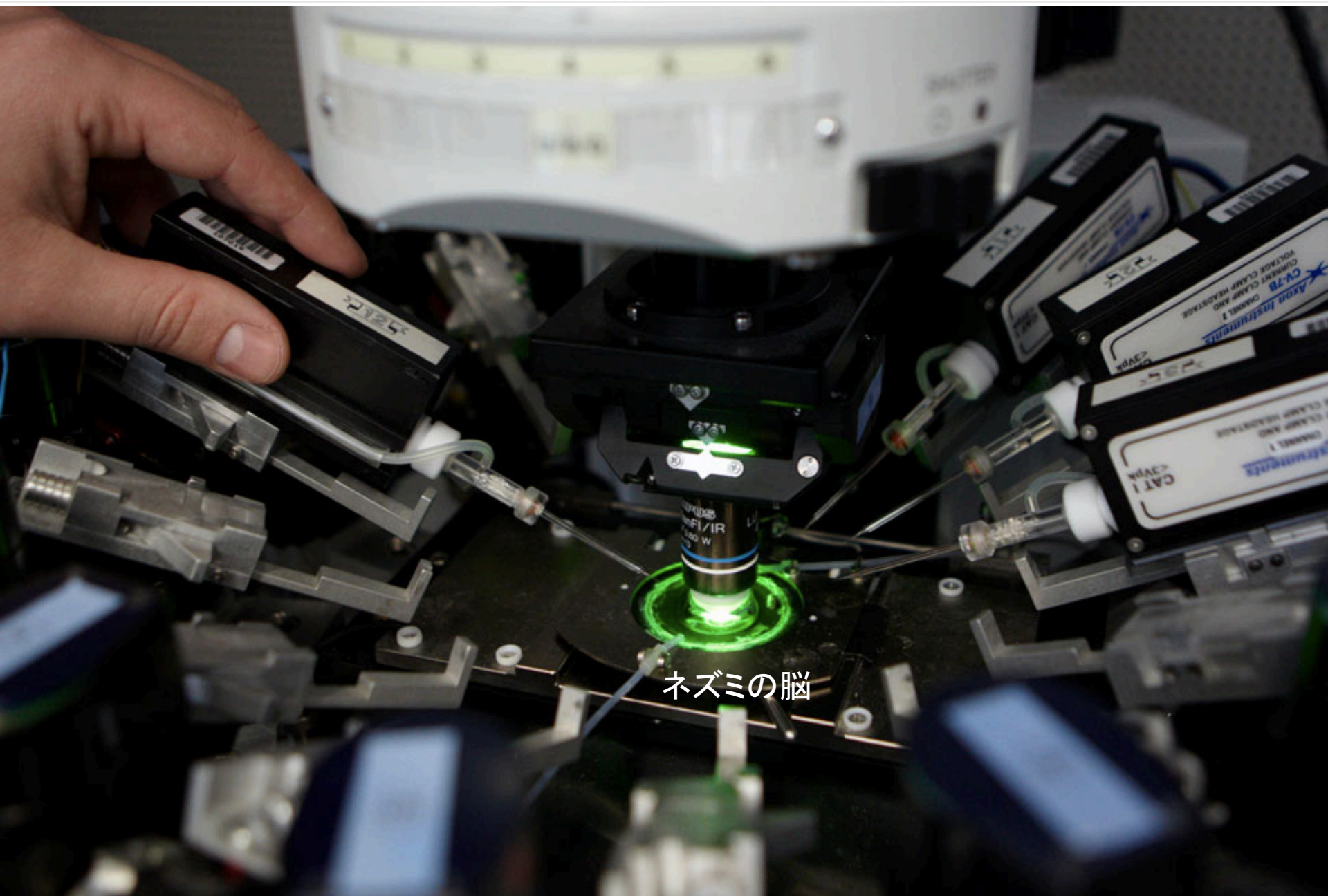
in vivo

in vitro

in silico



http://seedmagazine.com/content/article/out_of_the_blue/P1/



ネズミの脳



http://seedmagazine.com/content/article/out_of_the_blue/P1/

EU Human Brain Project



2013年10月 10年間で総額約12億ユーロ

EU Human Brain Project

ビジョン

- 人間の脳を理解することは、21世紀の科学が直面している最も偉大な挑戦の一つである。もしも、我々が、それに対して立ちあがることが出来るならば、我々は、我々を人間にしているものが何であるかについて深い洞察を得て、革命的なコンピュータ技術を構築し、脳の異常に対して新しい治療法を開発出来るだろう。今日、初めて、現代のICT技術が、こうした目標を到達可能なものにしている。

EU Human Brain Project

批判 ECへの公開質問状

Open message to the European Commission concerning the Human Brain Project

Sign the letter

Summary: Neuroscience advances our understanding of normal and pathological brain function, offering potentially enormous benefits to society. It is, therefore, critical to Europe. The Human Brain Project (HBP), sponsored by the European Commission (EC), was meant to forward this mission. However, due in great part to its narrow focus, it has been highly controversial and divisive within the European neuroscience community and even within the consortium, resulting in on-going losses of members. The HBP is now scheduled for review and we wish to draw the attention of the EC to these problems. We believe the HBP is not a well conceived or implemented project and that it is ill suited to be the centerpiece of European neuroscience. We are particularly concerned about the plan to tie a substantial portion European member states' neuroscience funding to the HBP through so-called 'partnering projects'. We call for the EC to go beyond the strict requirements of the upcoming review, to demand transparency and accountability and, if necessary, change the structure of the HBP's governance and supervision to correct their shortcomings. Failing that we call for the EC to redirect the HBP funding to smaller investigator-driven neuroscience grants. We stand fully behind a strong and united European neuroscience strategy and we pledge not to seek funding through HBP partnering projects that would compromise that mission.

[Official Response by the EC - 18 July 2014](#) | [Press Releases](#)

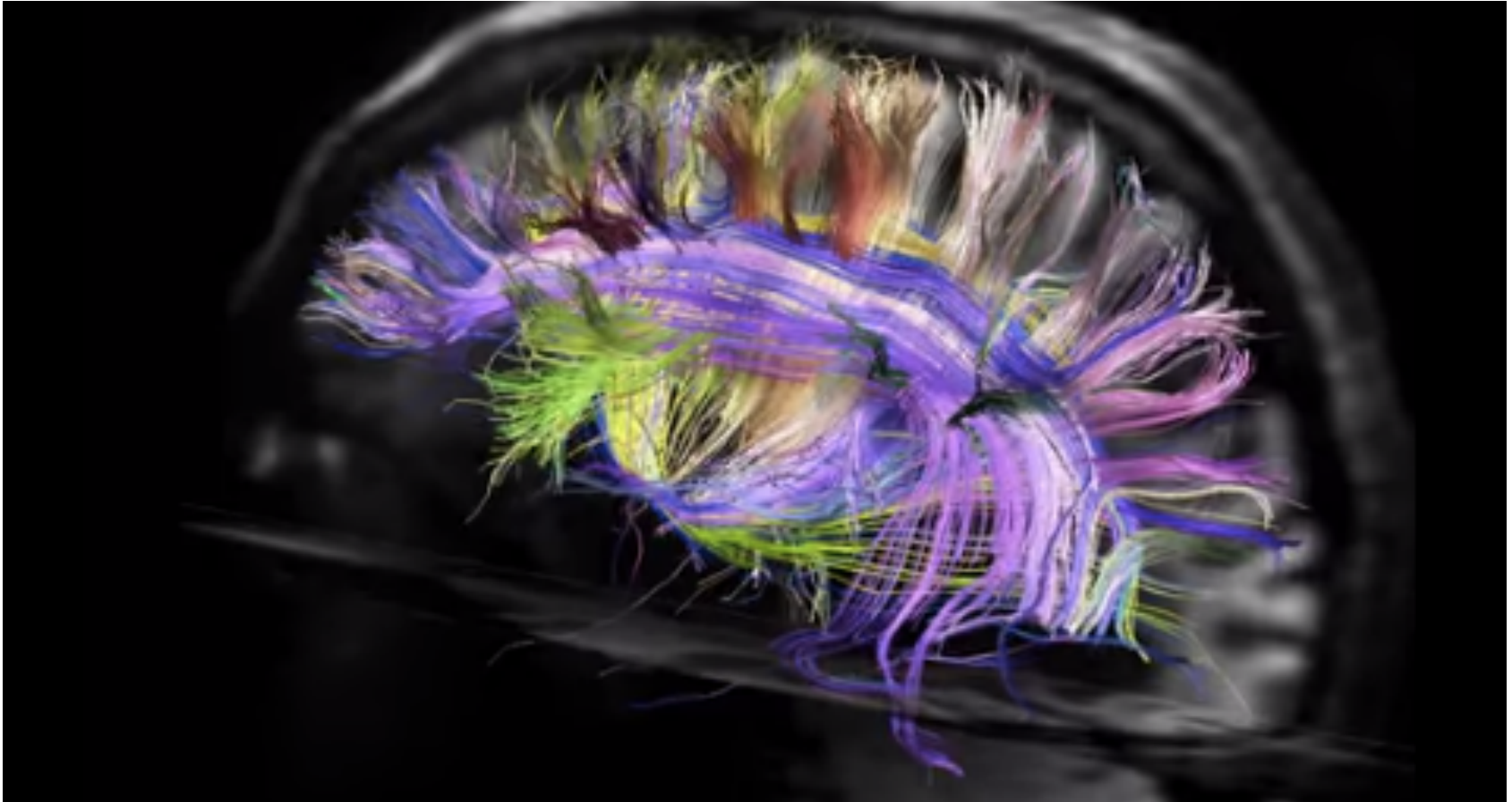
Read the full letter

脳研究をめぐる「対立」の構図

認知科学 vs 脳科学、トップダウン vs ボトムアップ

- 発端は、次年度のプロジェクトの予算配分で、認知科学的なアプローチの予算が、ばっさりと切られたことにあるらしい。それに反発した研究者のグループが、150名の連名で、欧州委員会に公開質問状を提出した。
- 対立の根底にあるのは、脳研究でのアプローチの違い。Henry Markramら主流派は、ニューロンとシナプスの数学的モデルに基づいて脳全体のモデルを**ボトムアップ**に作り上げようというアプローチ。一方、反対派は、脳研究には、認知科学の知見に基づいた**トップダウン**のリバーズ・エンジニア的なアプローチが必要だという。
- 「鳥の羽の全てをシミュレーションしたとしても、鳥が空を飛べることを解明出来ないのと同じことだ」

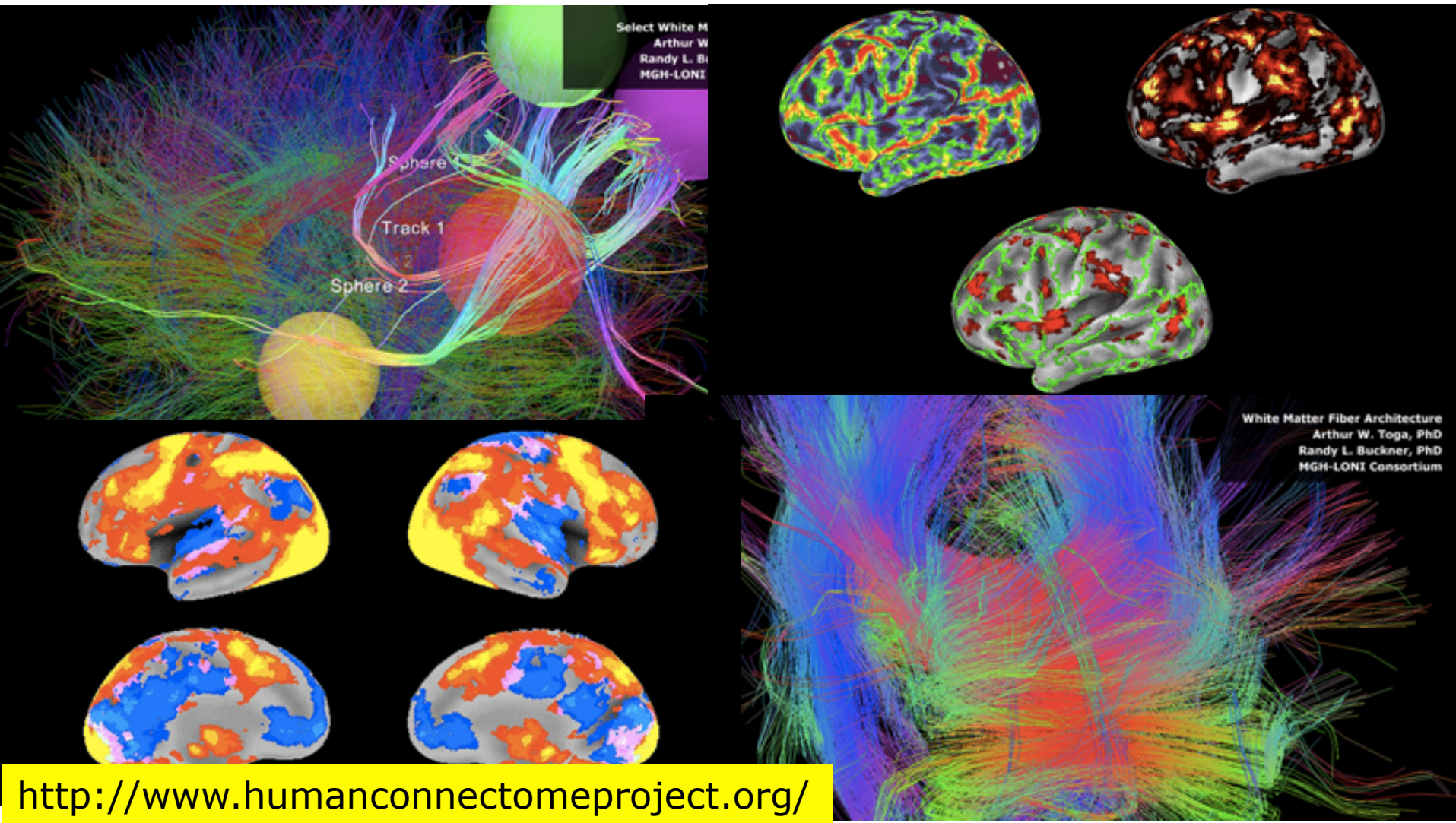
NIH Human Connectome Project



2010年10月 40億円の賞金

<http://www.nih.gov/news/health/sep2010/nimh-15.htm>

NIH Human Connectome Project





US BRAIN Initiative NIHのもとで、12年間で4,500億円投資する



2014/06/05

<http://1.usa.gov/1pIIhvx>



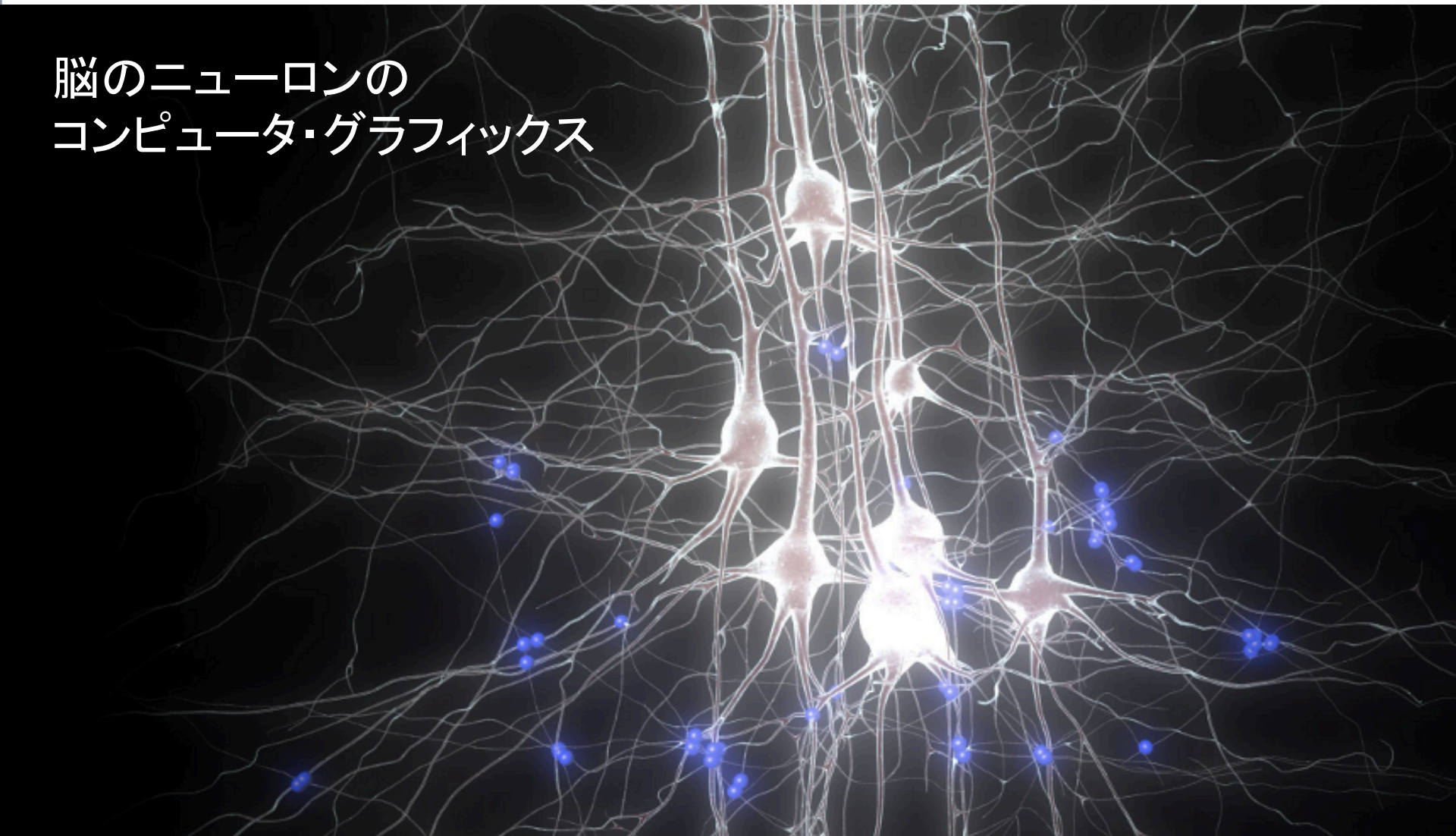
ニューロンの働き

ニューロンの働き

脳の働きも、最終的には、脳を構成する無数のニューロンの働きに帰着する。

コンピューター上のニューラル・ネットワークを構成する個々の「ニューロン」の動作原理は、生体のニューロンの働きをシミュレートしたものである。

脳のニューロンの コンピュータ・グラフィックス



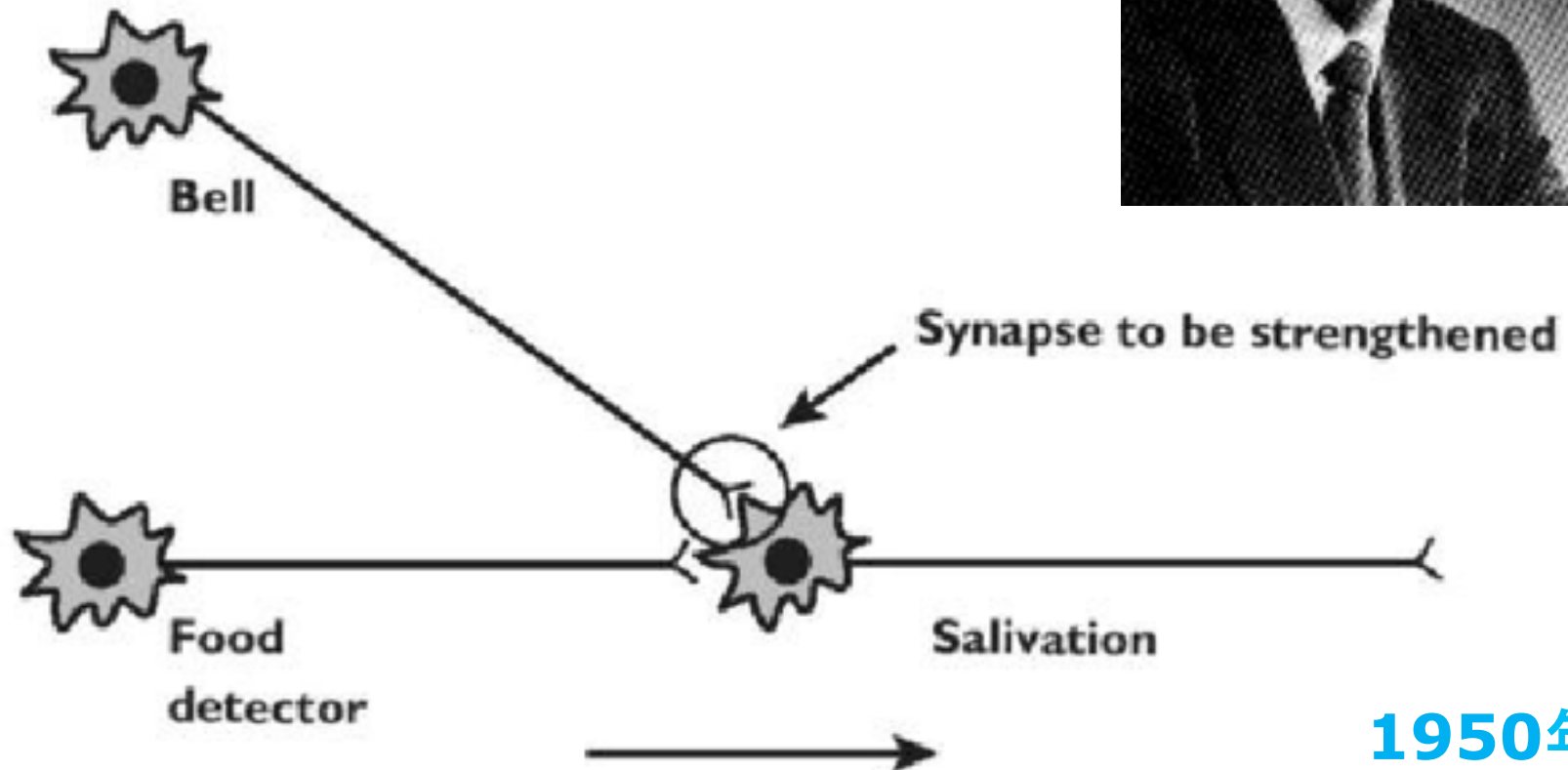
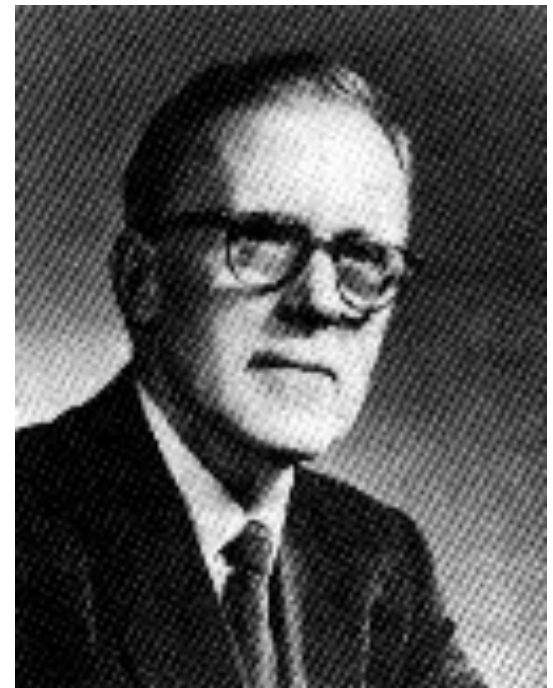
ヘッブの法則

Donald O. Hebb

- ヘッブの法則(ヘッブのほうそく)は、脳のシナプス可塑性についての法則である。ヘッブ則、ヘブ則とも呼ばれる。心理学者のドナルド・ヘッブによって提唱された。ニューロン間の接合部であるシナプスにおいて、シナプス前ニューロンの繰り返し発火によってシナプス後ニューロンに発火が起こると、そのシナプスの伝達効率が增強される。また逆に、発火が長期間起こらないと、そのシナプスの伝達効率は減退するというものである。
- *The Organization of Behavior*. 1949年

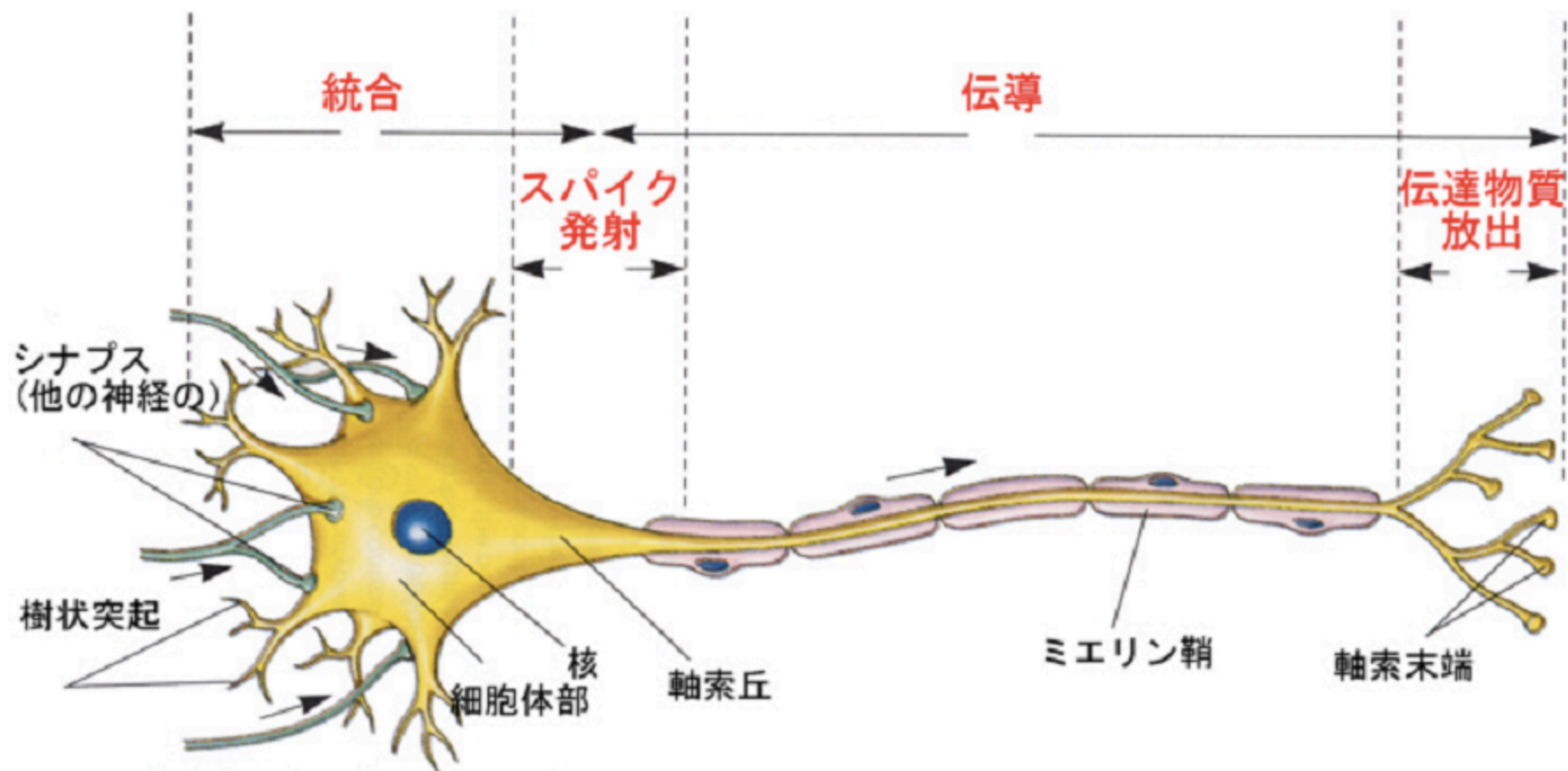
ヘッブの法則

Donald O. Hebb



1950年代

神経細胞の仕組み

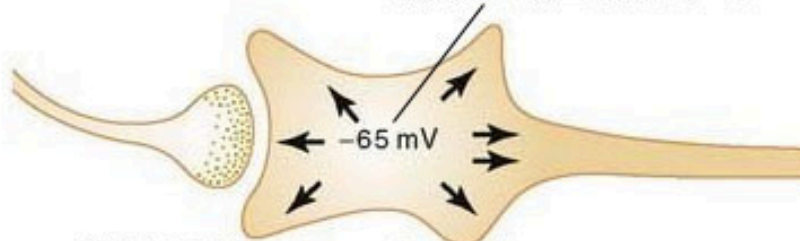


別の神経細胞からシナプスを通して神経伝達物質を受け取る。
刺激が強いと神経細胞は興奮し、スパイクという電気を流す。

興奮性シナプスと抑制性シナプス

シナプスによる興奮vs抑制

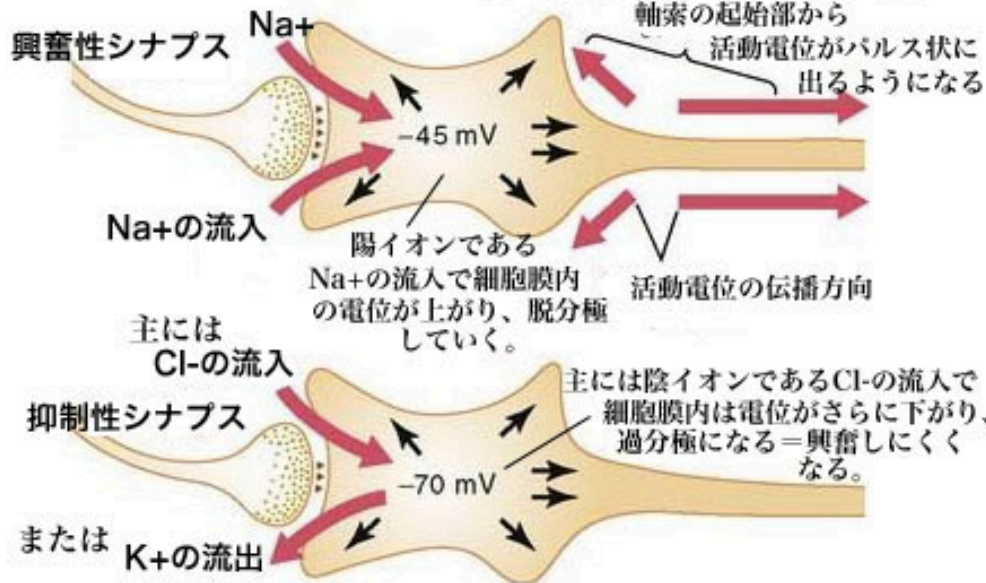
細胞体での静止電位はだいたい-60mV



興奮も抑制もされていない普通のニューロン

ニューロンがある程度以上興奮すると
軸索の起始部から

活動電位がパルス状に
出るようになる

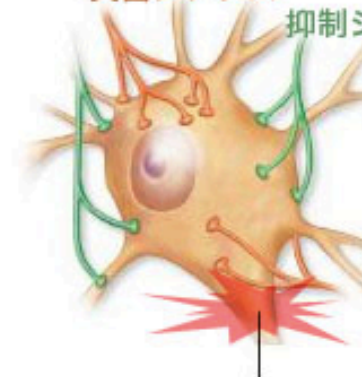


Guyton & Hall Medical Physiologyより

Ravenら Biologyより

興奮シナプス

抑制シナプス



軸索の起始部

(興奮の合計) - (抑制の合計) > 閾値
となると、興奮の大きさに合わせた頻度で
パルス状に活動電位を発生し続ける。

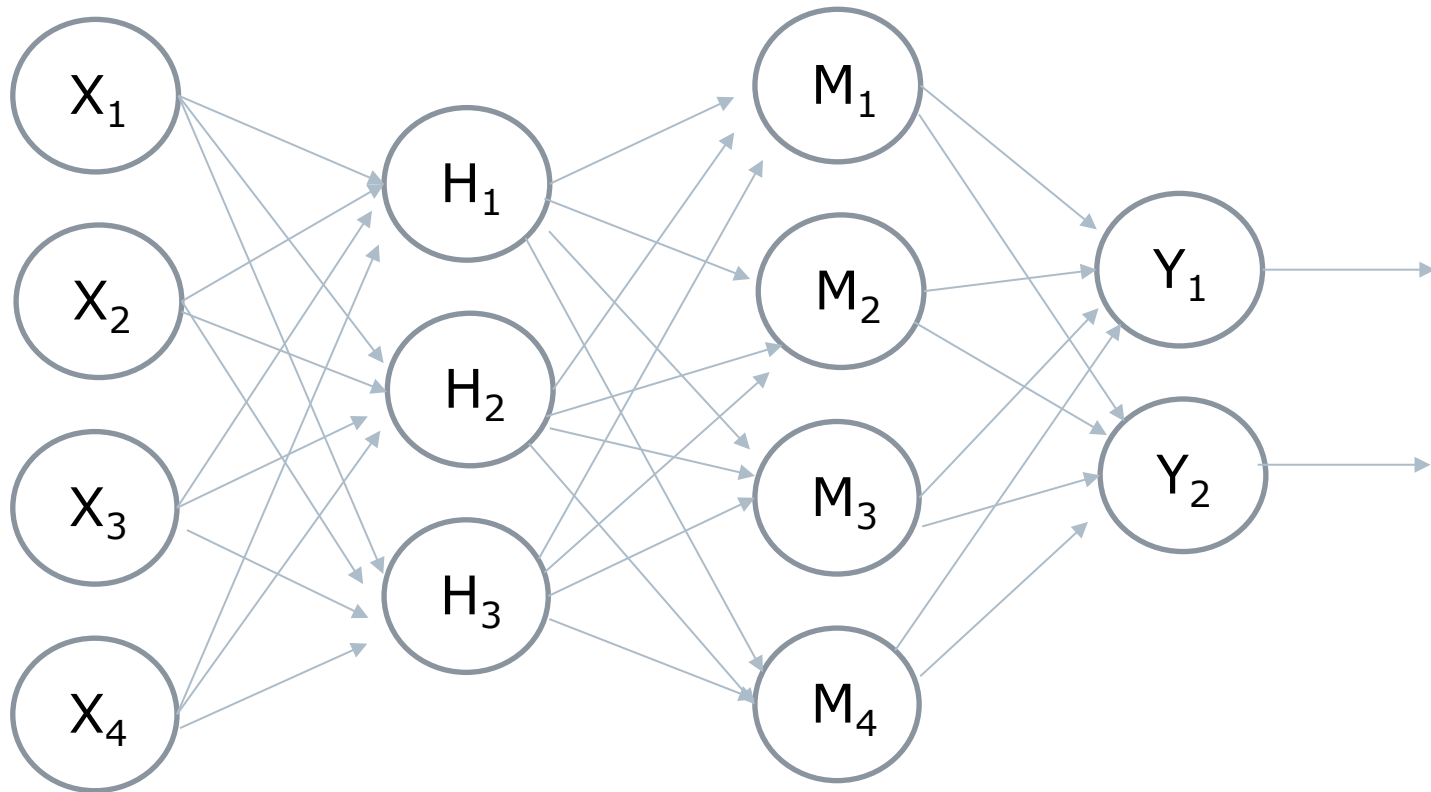


ニューラル・ネットワークとその表現

ニューラル・ネットワークとその表現(1)

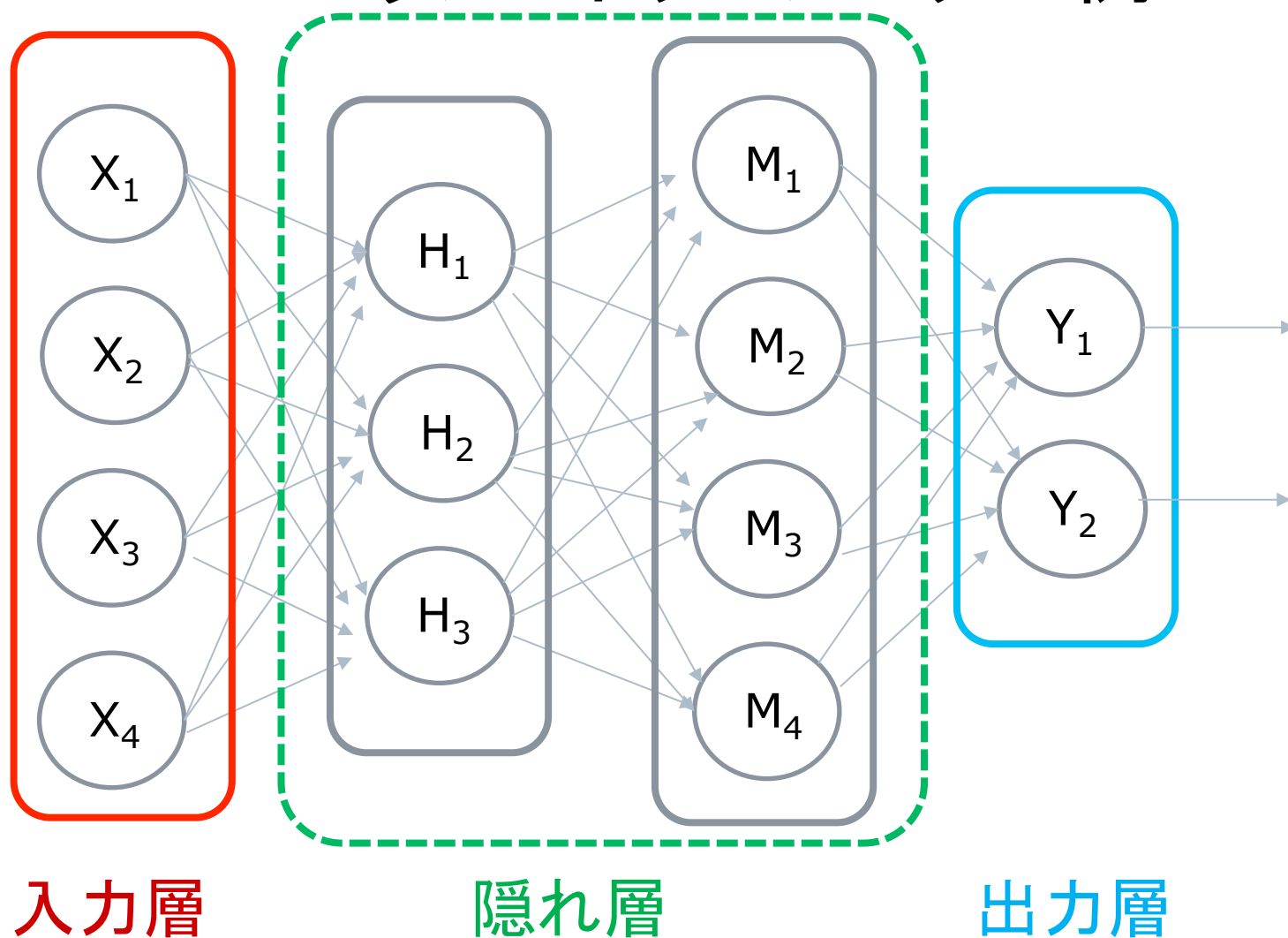
- ニューロンは、いつ発火するのか？
- 一つのニューロンの動きを考える
- 一つのニューロンの内部の状態の表現
- 複数のニューロンからなる一つの層の内部の状態の表現
- 式は便利だ！ $Y = W \cdot X + b$ が表すもの

ニューラル・ネットワークの例



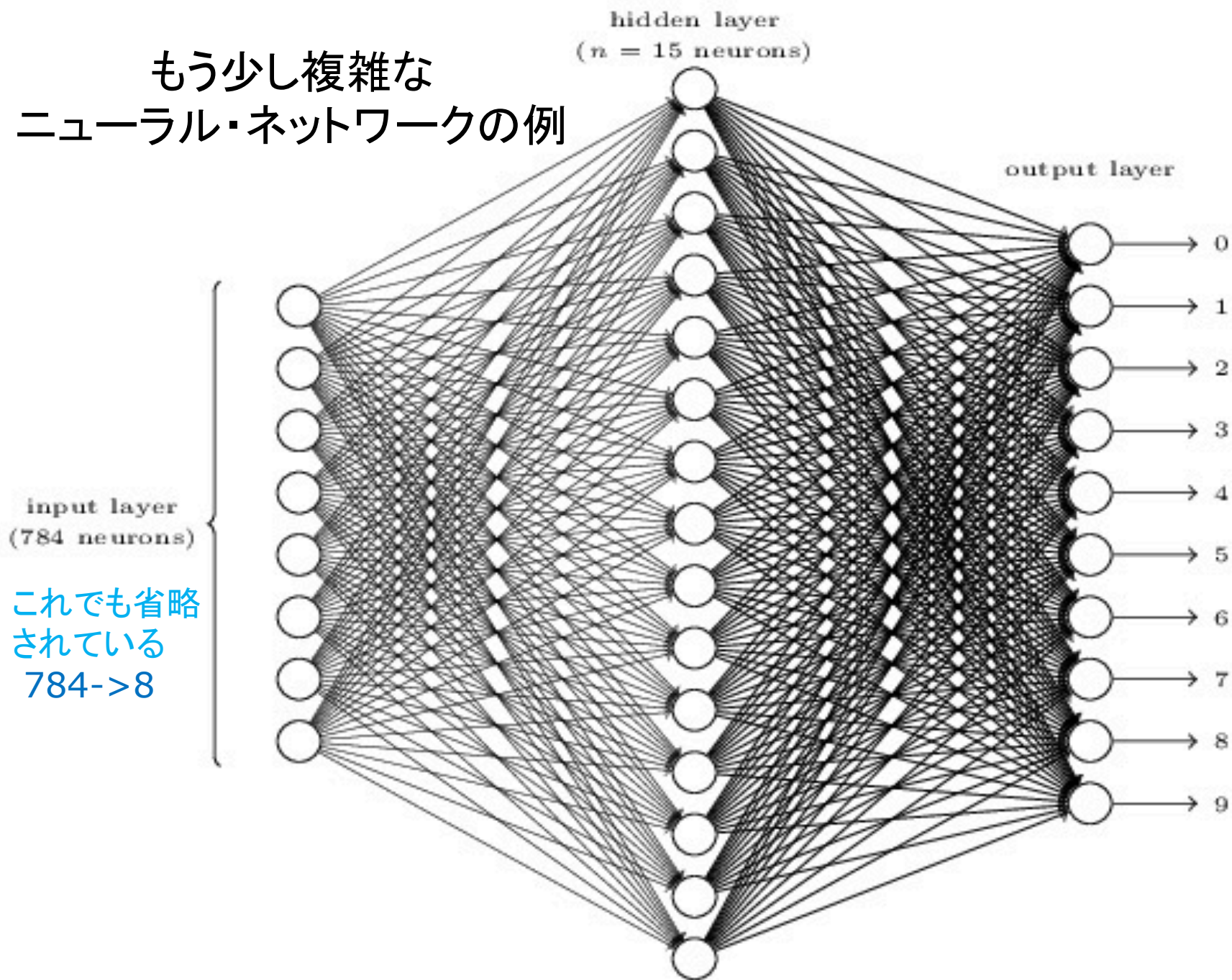
一つ一つの丸が、一個のニューロンを表している。
一つのニューロンは、その「出力」を、他のニューロンに、送り出す。

ニューラル・ネットワークの例



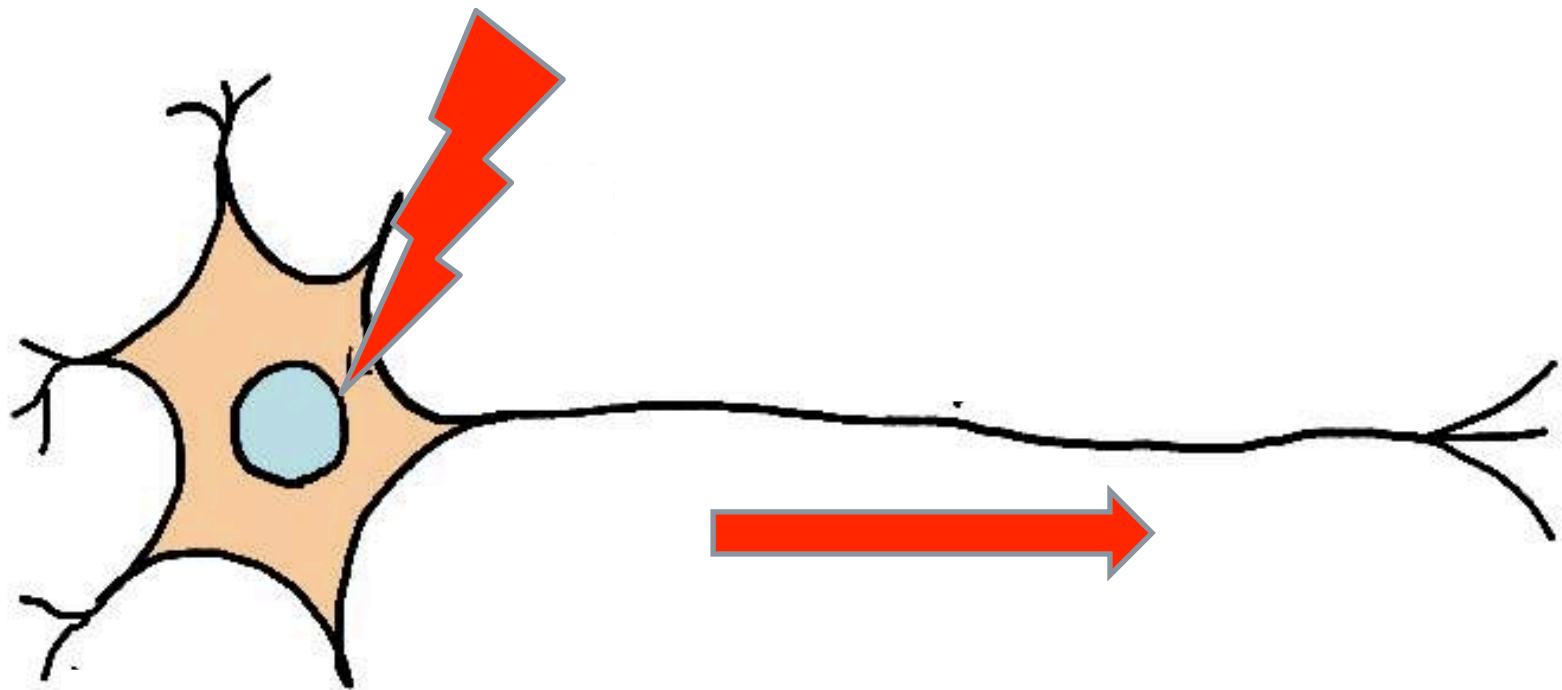
複数のニューロンの集まりが、「層」をなしている。

もう少し複雑な ニューラル・ネットワークの例

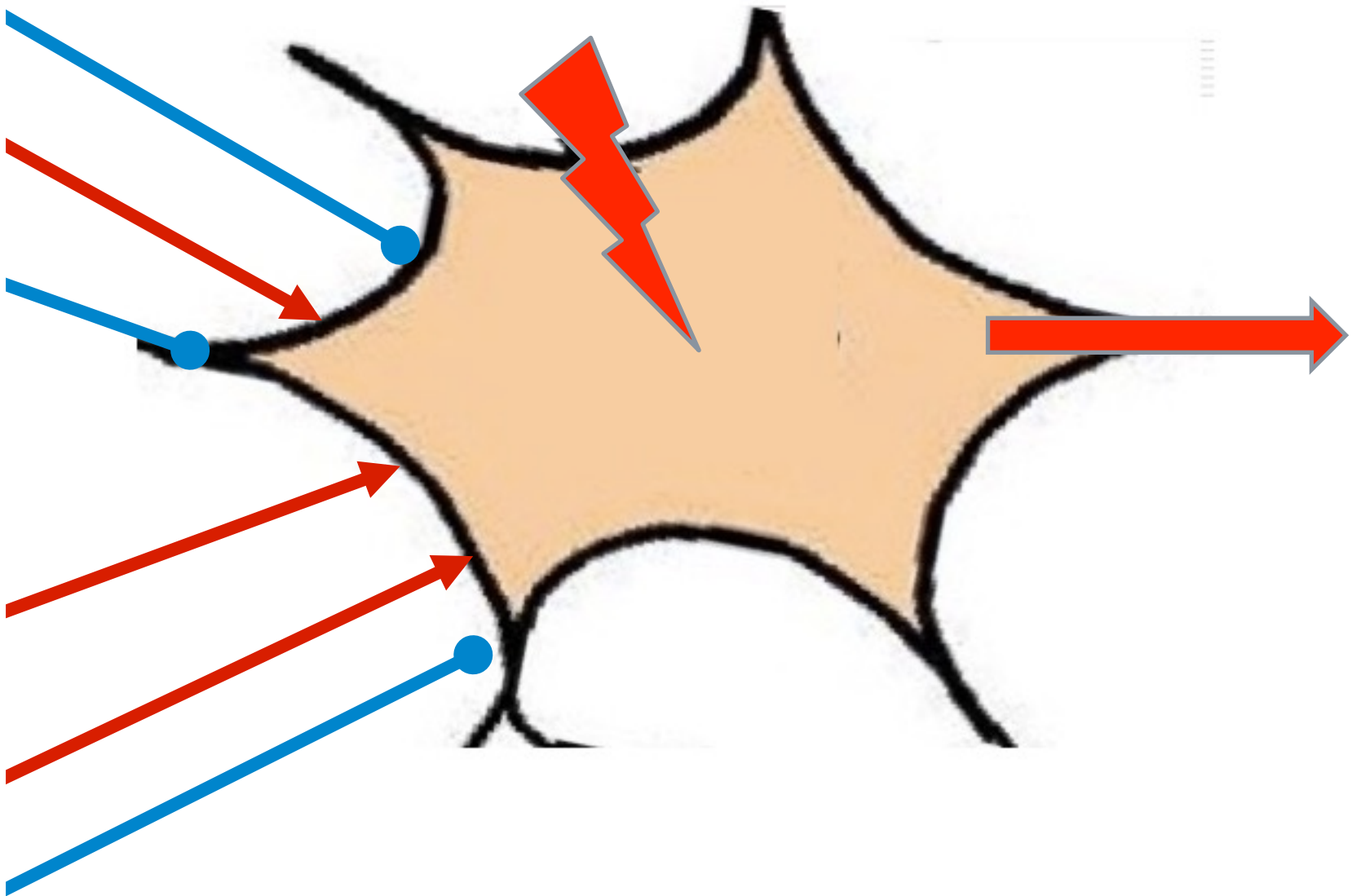


ニューロンは、いつ発火するのか？

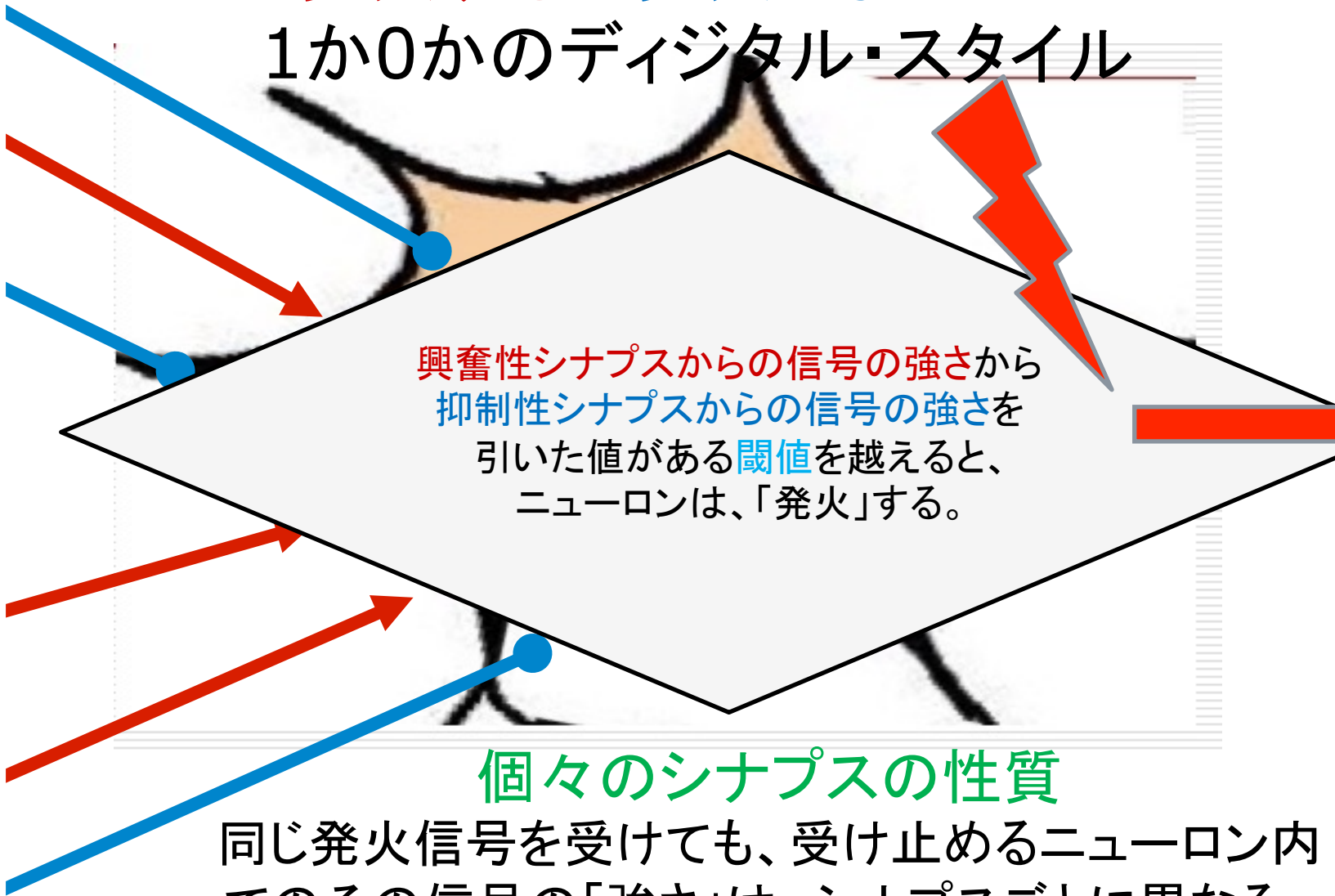
ニューロンは、
いつ発火するのか？



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



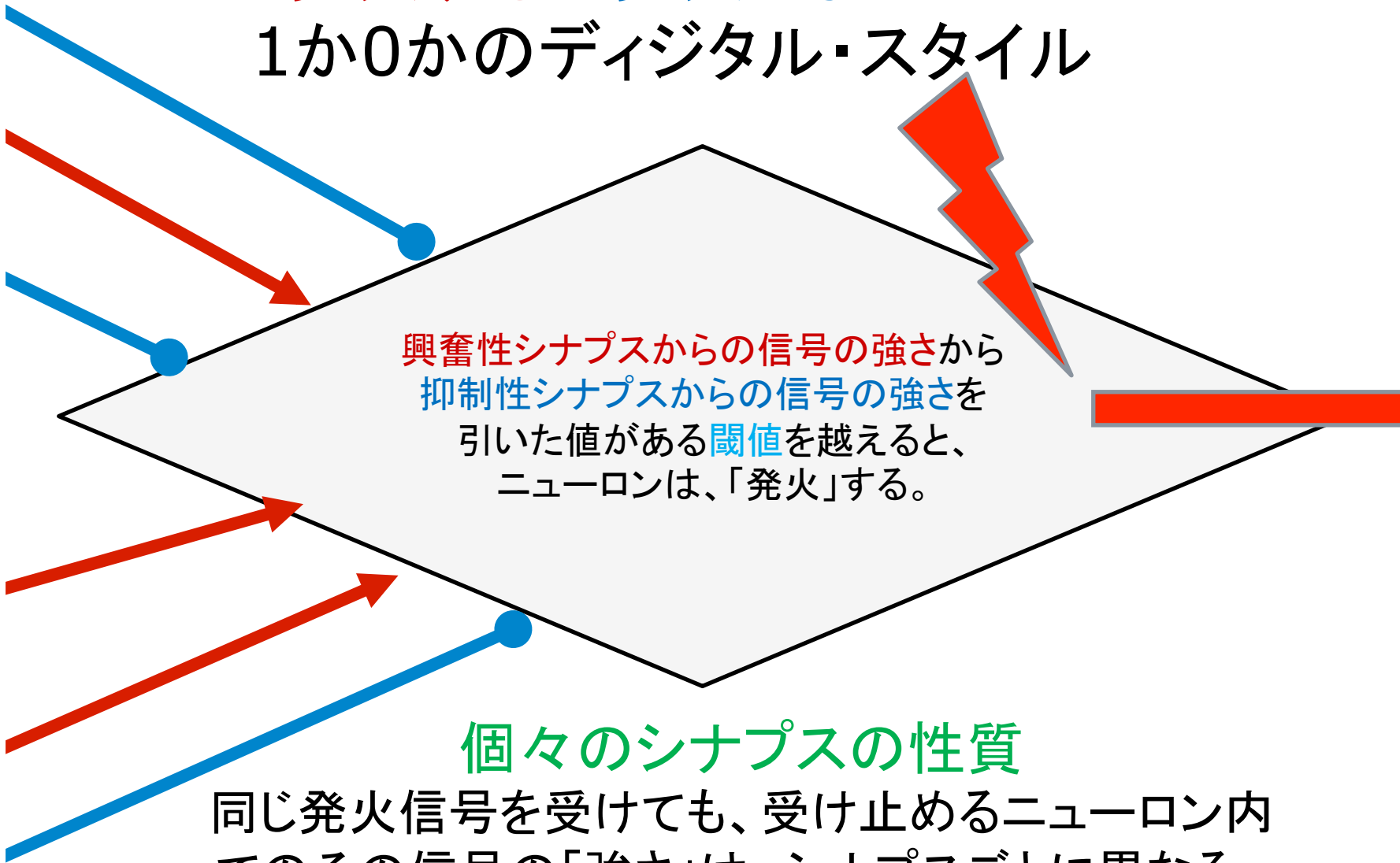
興奮性シナプスからの信号の強さから
抑制性シナプスからの信号の強さを
引いた値がある閾値を越えると、
ニューロンは、「発火」する。

The diagram shows a central gray diamond representing a 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 side 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 arrows (blue and red) point towards the diamond from the left, representing synaptic inputs. Some of these arrows terminate at blue circular nodes on the diamond's boundary. A large red lightning bolt strikes the right side of the diamond, symbolizing an action potential. To the right of the diamond, a thick red horizontal bar represents the output signal. Inside the diamond, text explains the firing process based on the balance of excitatory and inhibitory inputs relative to a threshold.

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: $3+3+2=7$

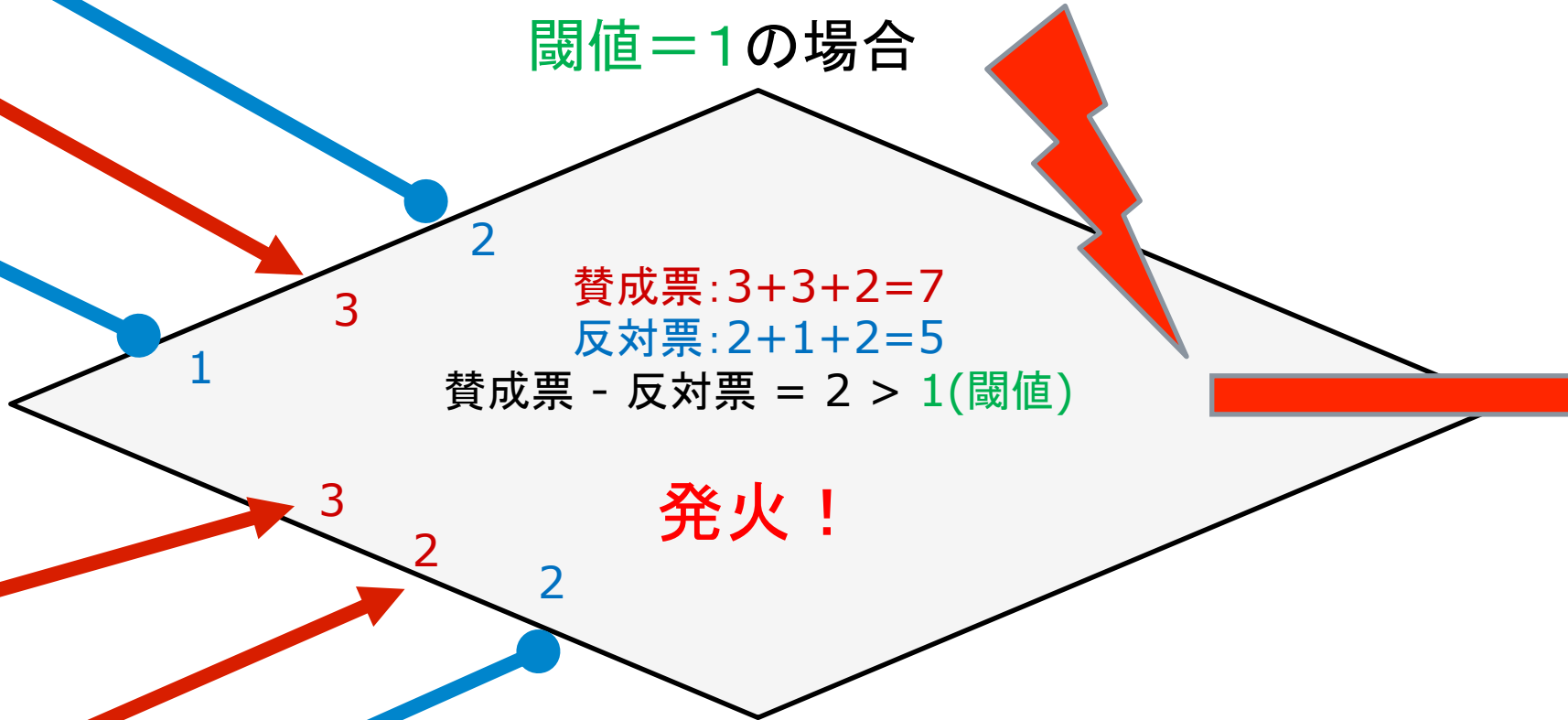
反対票: $2+1+2=5$

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

X1=1

X2=0

X3=1

X4=0

X5=1

X6=1

1

2

3

賛成票: $0+0+2=2$

反対票: $2+1+2=5$

賛成票 - 反対票 = $-3 < 1$ (閾値)

発火せず！

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

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」

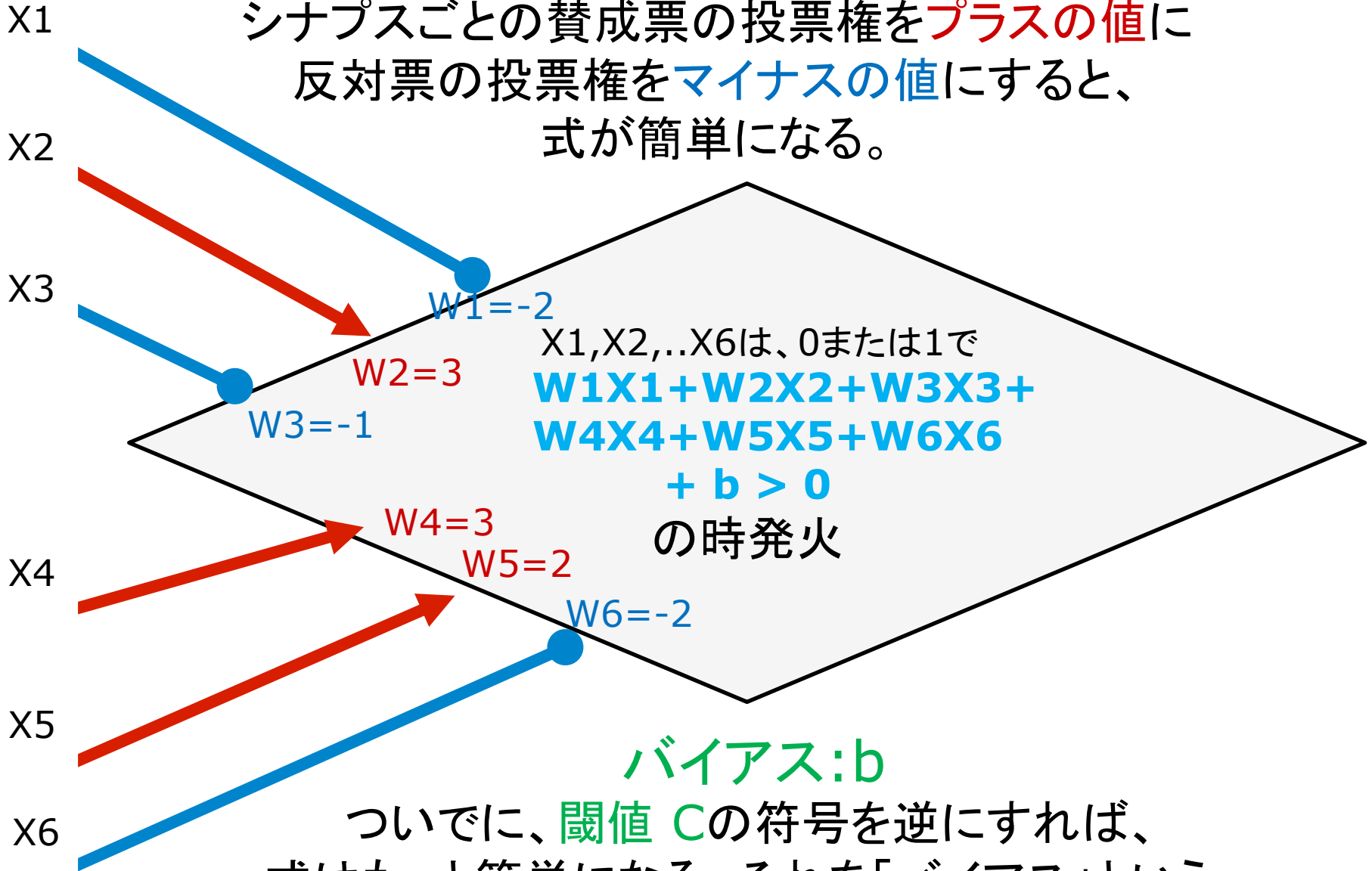
- 賛成票の数 A をプラスの値に
反対票の数 B をマイナスの値 B' にし、
閾値 C の符号を逆にして C' (バイアス) とすると、
発火の条件 $A - B > C$ (すなわち、 $A - B - C > 0$) は、
 $A + B' + C' > 0$ の形になる。
- 各シナプスごとに異なる投票権の大きさを「重み」という。
興奮性シナプスはプラスの「重み」を持ち、抑圧性シナプスはマイナスの「重み」を持つと考えるということ。
- 一つの入力 X_i (0か1の値をとる) には、それに対応した一つのシナプスが存在する。そのシナプスの重みを W_i とすれば、この一つのシナプスの投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

$X_1 = 1$

$X_2 = 0$

$X_3 = 1$

$X_4 = 0$

$X_5 = 1$

$X_6 = 1$

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

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

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

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

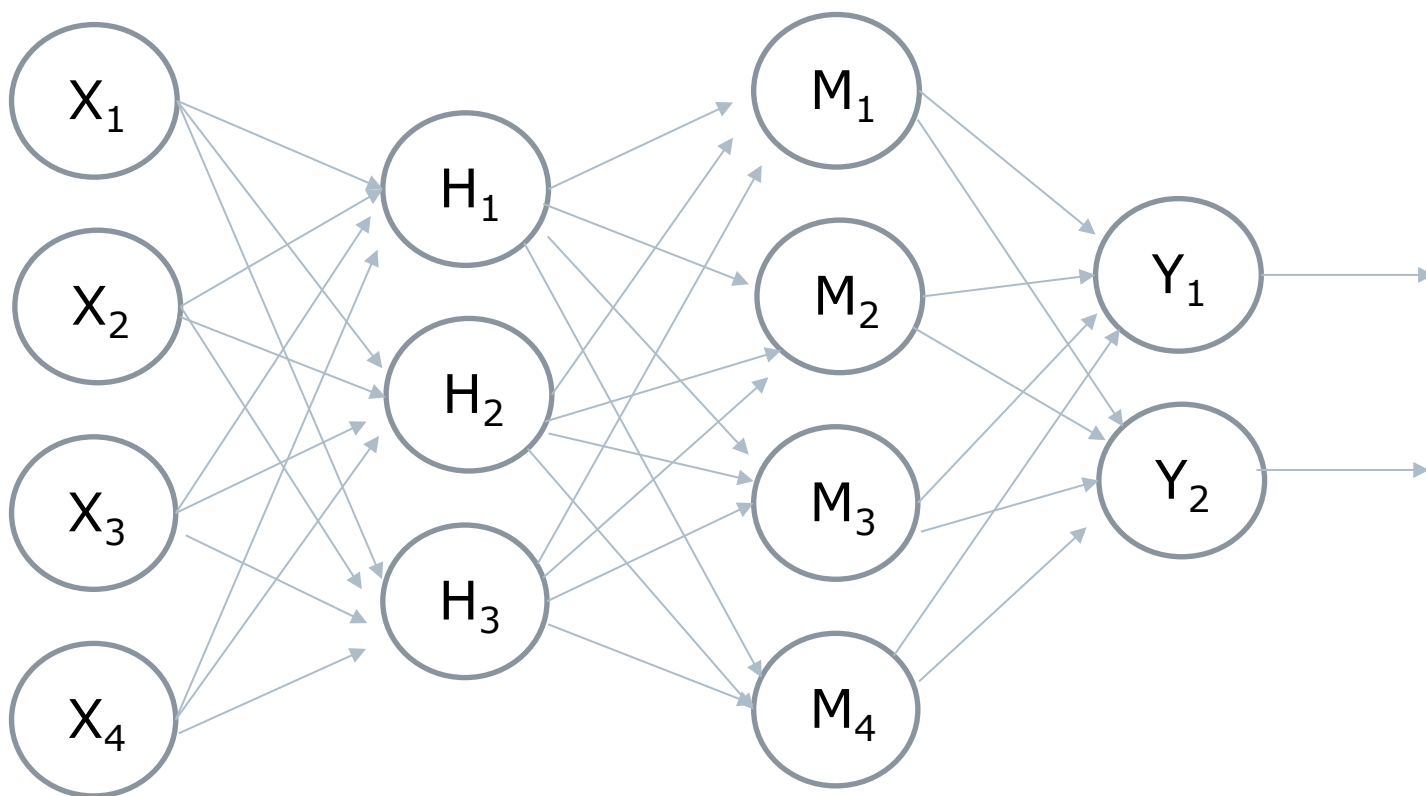
2

-2

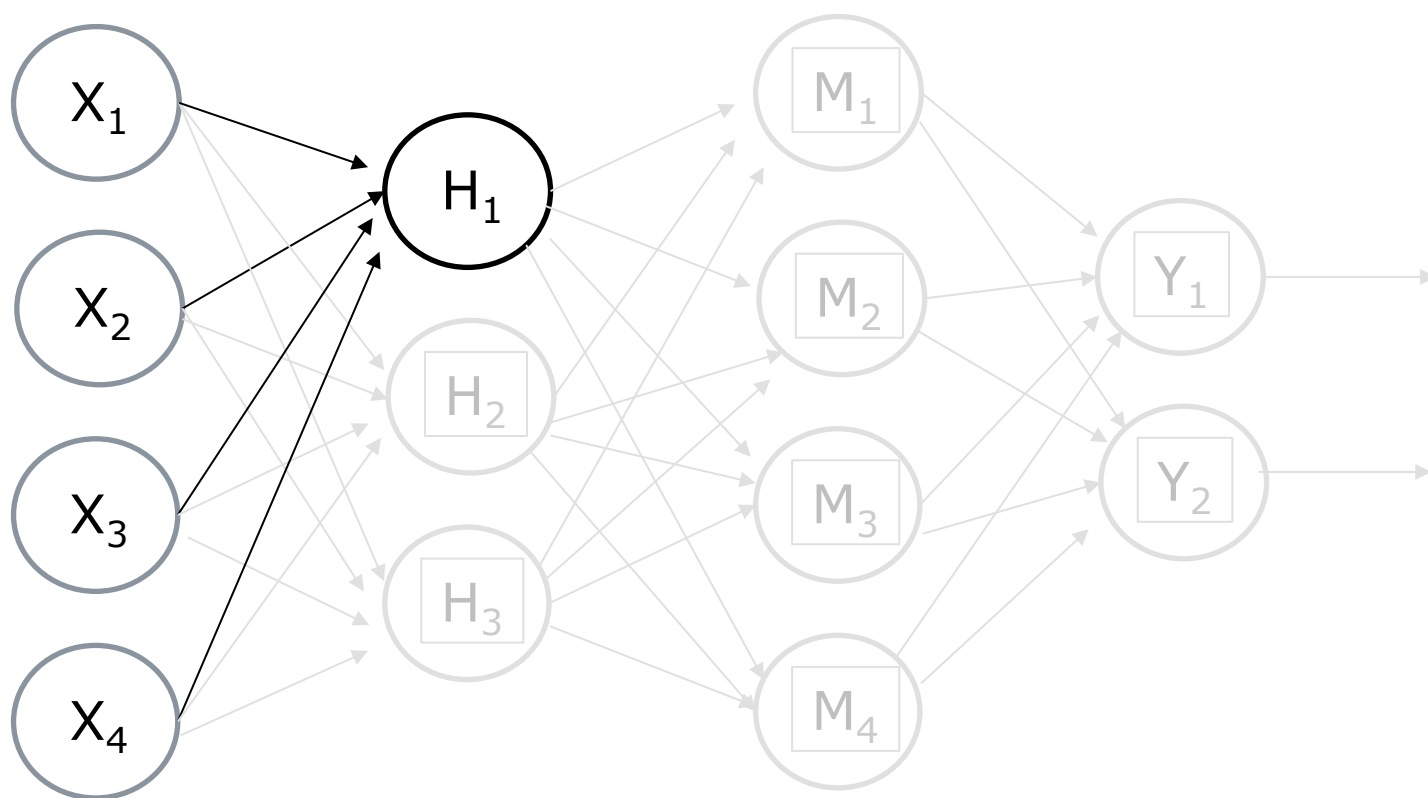
一つのニューロンの 内部の状態の表現

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。

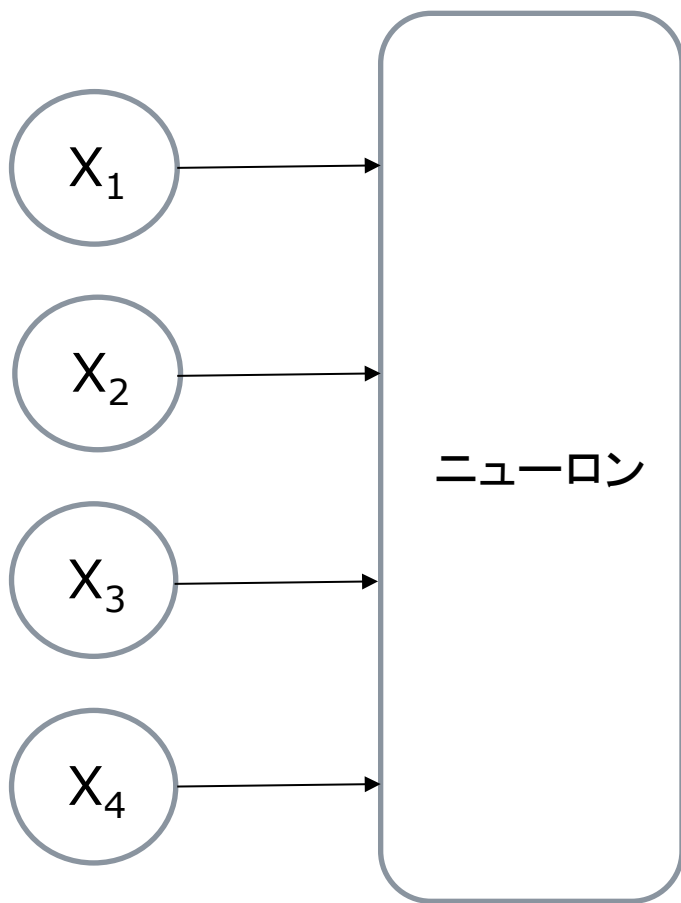
Full Connectな ニューラル・ネットワーク



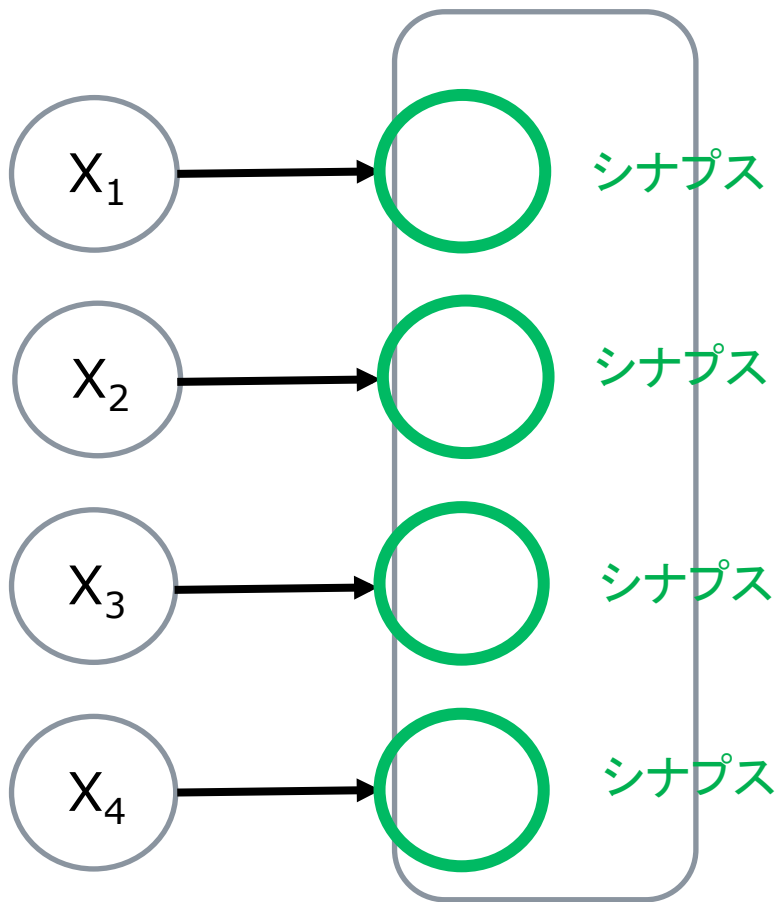
入力を受け取る一つのニューロンを見る



入力を受け取る一つのニューロン

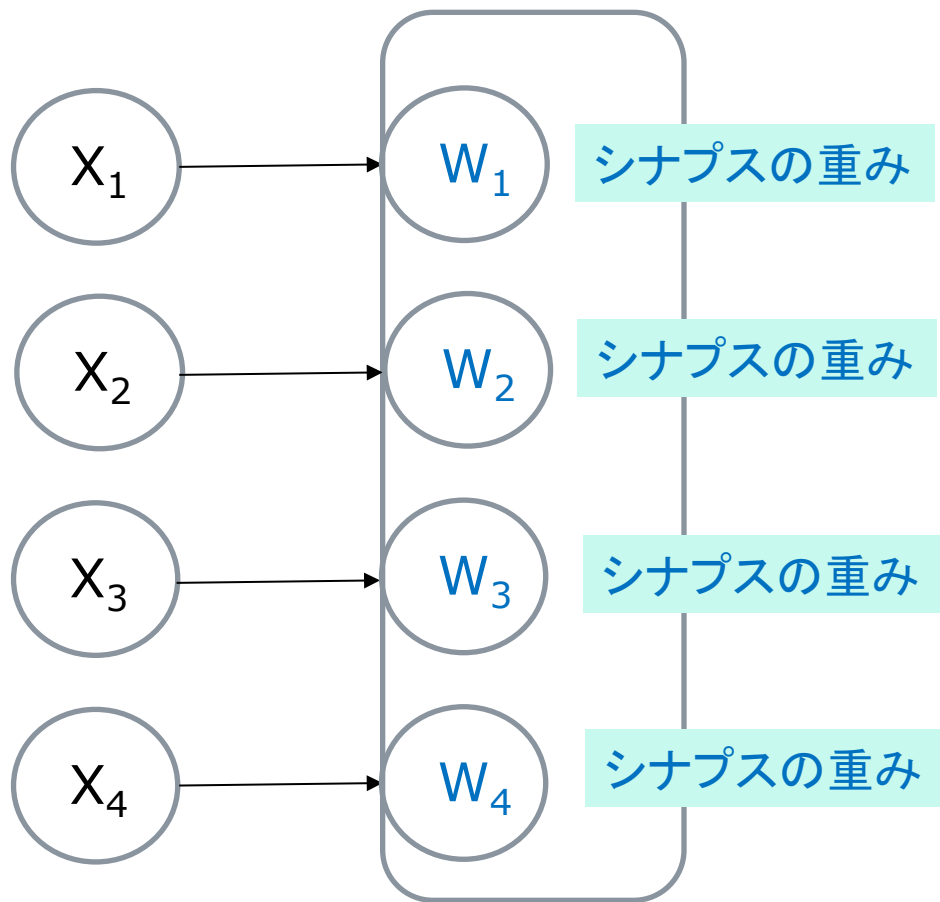


ニューロンは、シナプスを通じて入力を受け取る



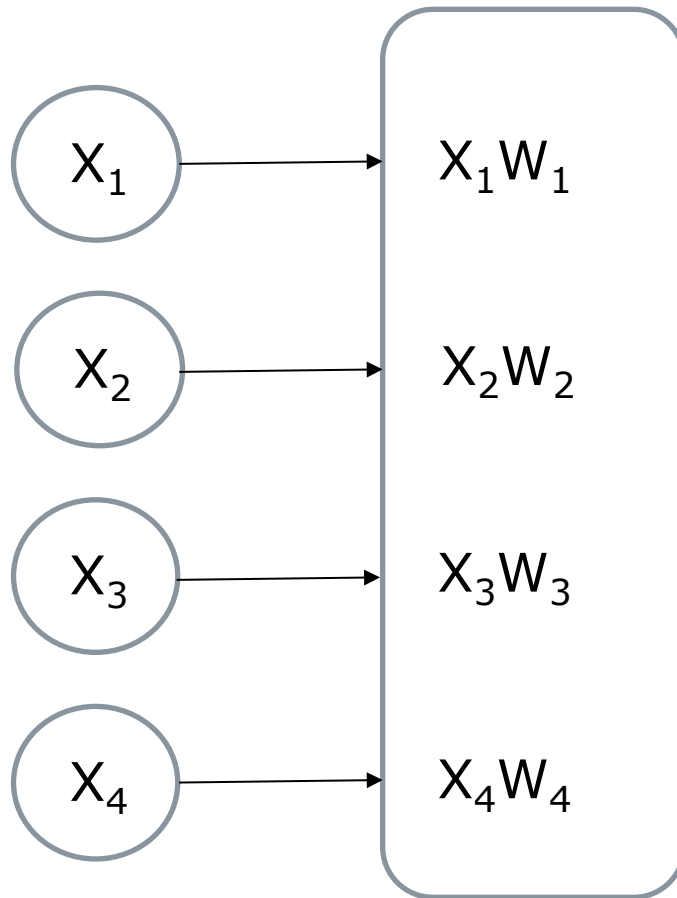
シナプスの数は、
入力の数に等しい

シナプスには、「**重み**」がある



重みの数は、
シナプスの数、
すなわち
入力の数に等しい

「多数決」で決めよう。入力は、シナプスの重みを掛けられてニューロンに送られる



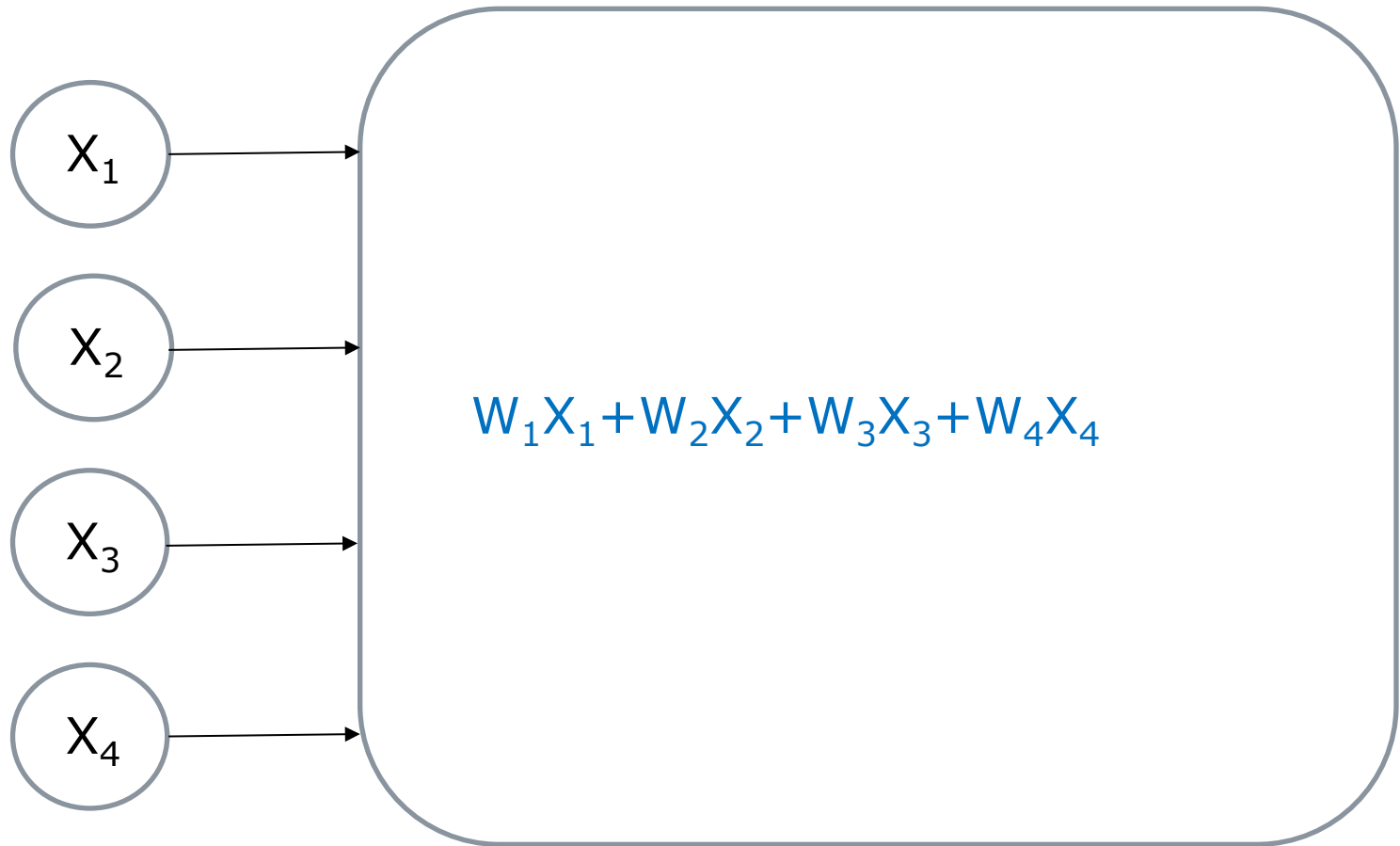
ニューロンが出力をするか否かは、シナプスの「多数決」で決められる。

ただし、すべてのシナプスが、平等な一票を持つわけではない。シナプスは、その「重み」に応じた投票権を持つ。株主総会で、持ち株数に応じた投票権を持つのと同じである。

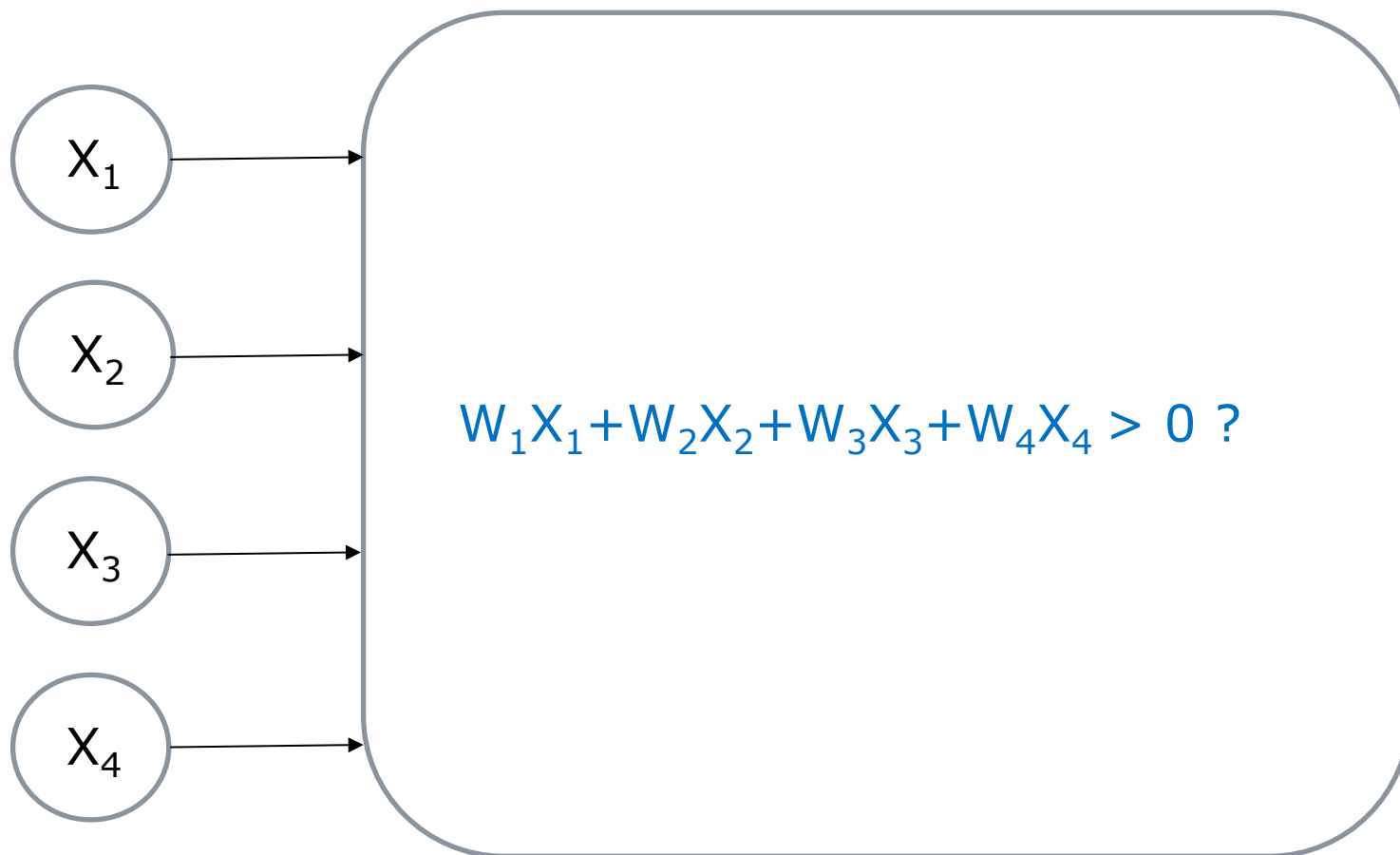
シナプス i が、重み W_i を持つ時、入力 X_i に対して、重みを掛けて、 $X_i \cdot W_i$ の投票を行う。

プラスの重みを出力賛成
マイナスの重みを出力反対
と、考えよう。

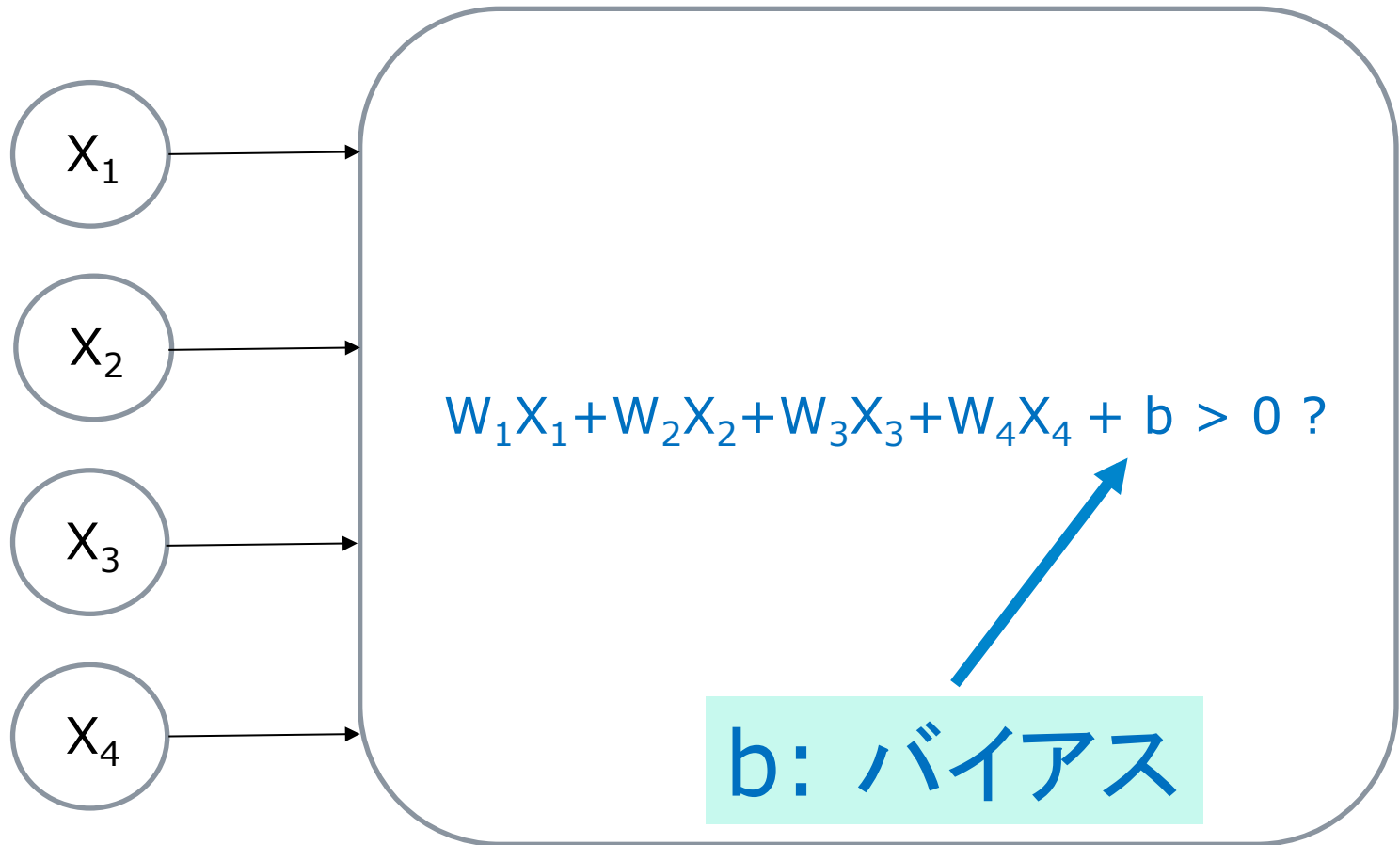
重みづけられた入力、すなわち投票を、
すべて、足し合わす。これが、投票結果。



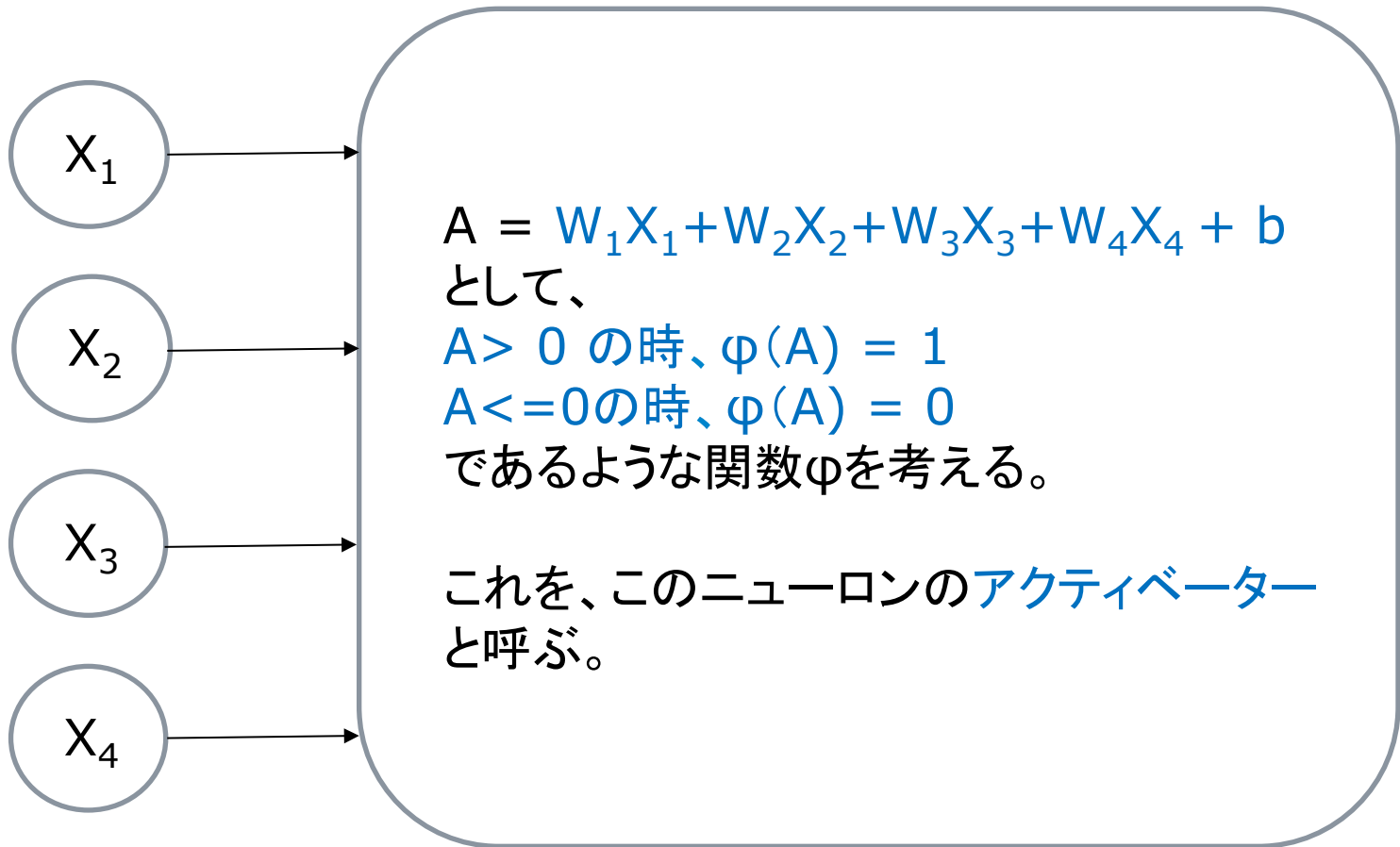
投票結果がゼロより大きかったら、ニューロンは「発火」するのか？ そうではない。



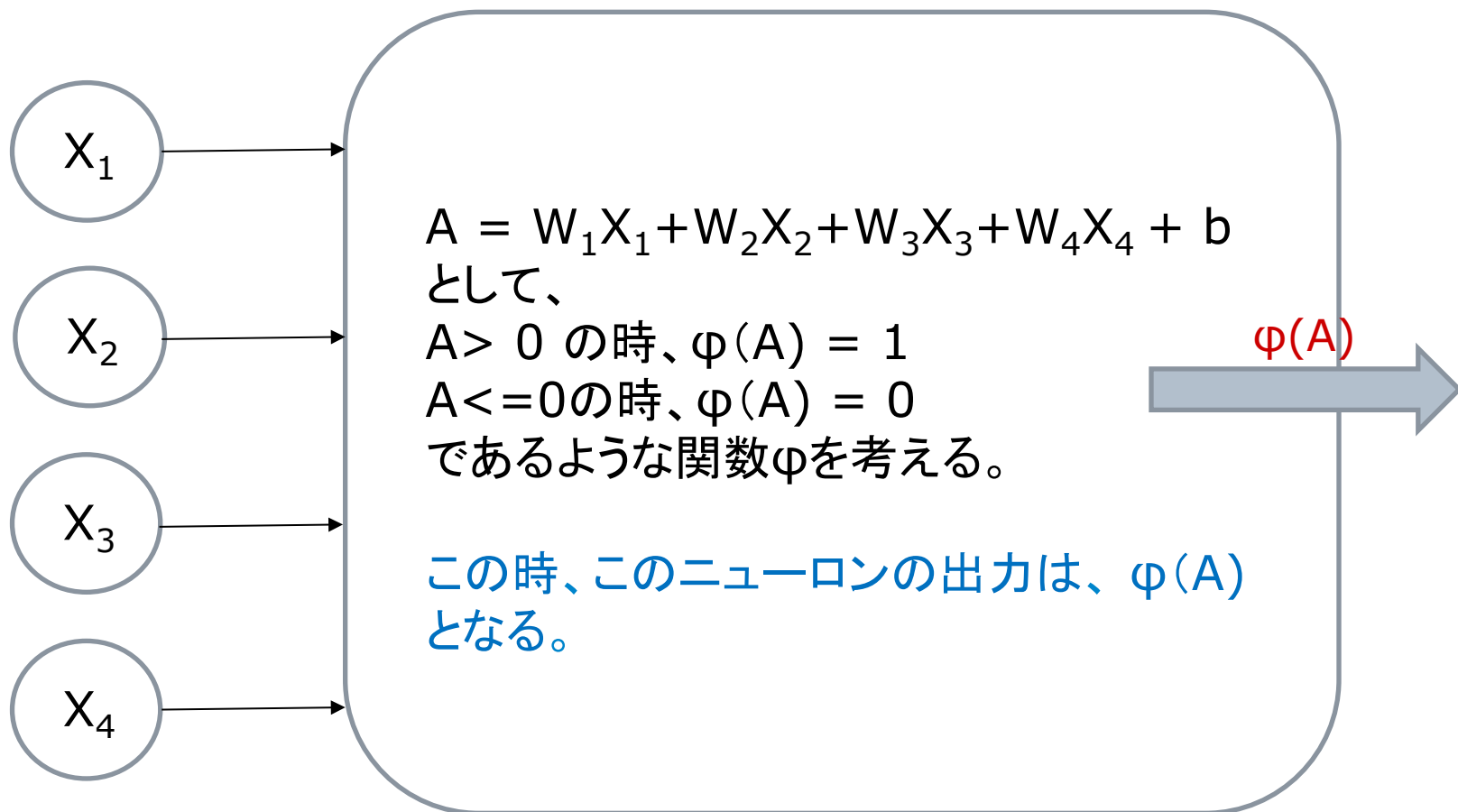
「発火」の条件には、ある閾値がある。それを**バイアス**という。投票結果とバイアスを合わせたものがゼロを超えた時、ニューロンは発火する



アクティベーター



入力を出力に変える: アクティベーター



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

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

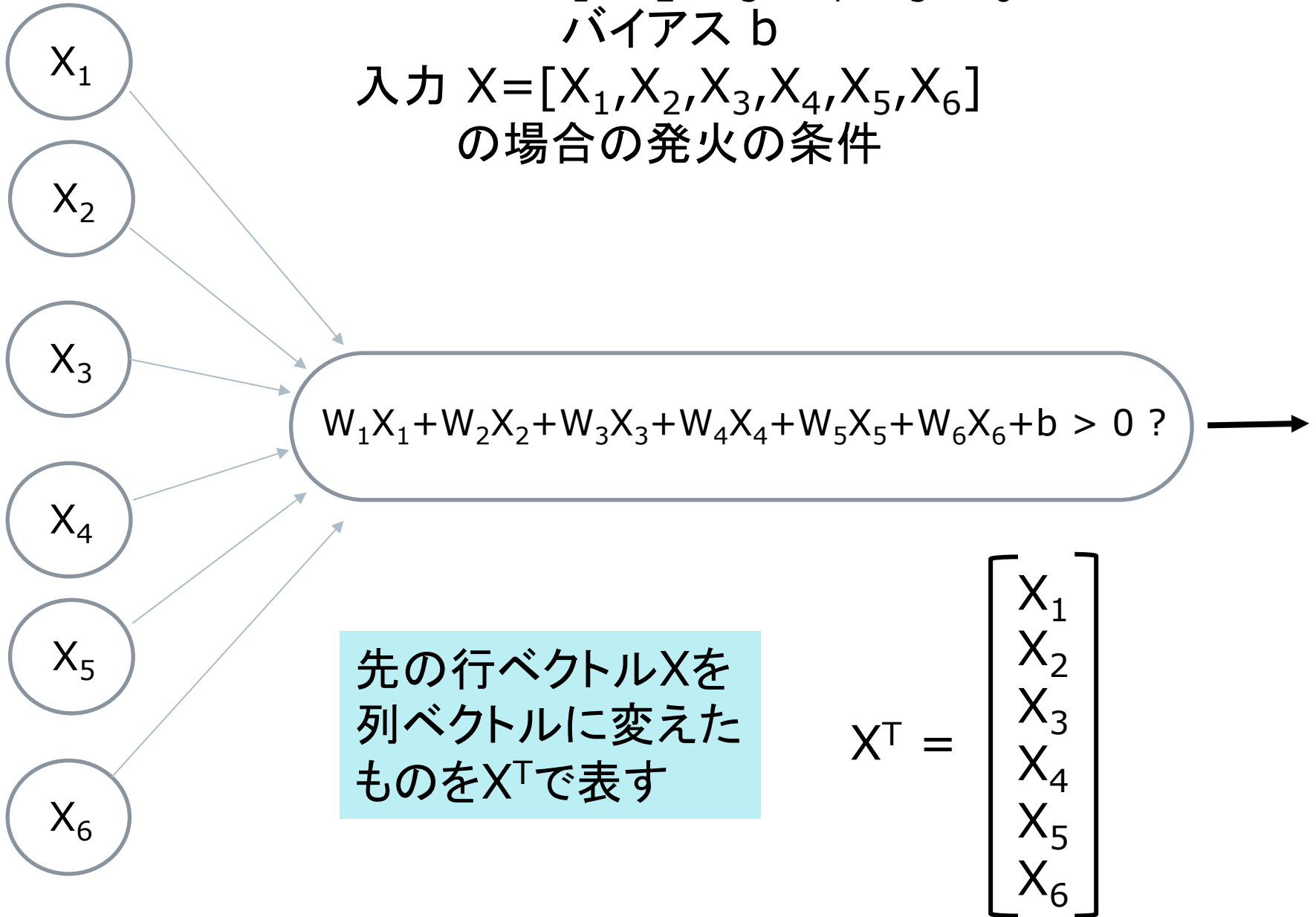


重みと入力の対応する項目同士を
かけて、すべて足し合わせて、それに
バイアス b を足したもの。

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。~~

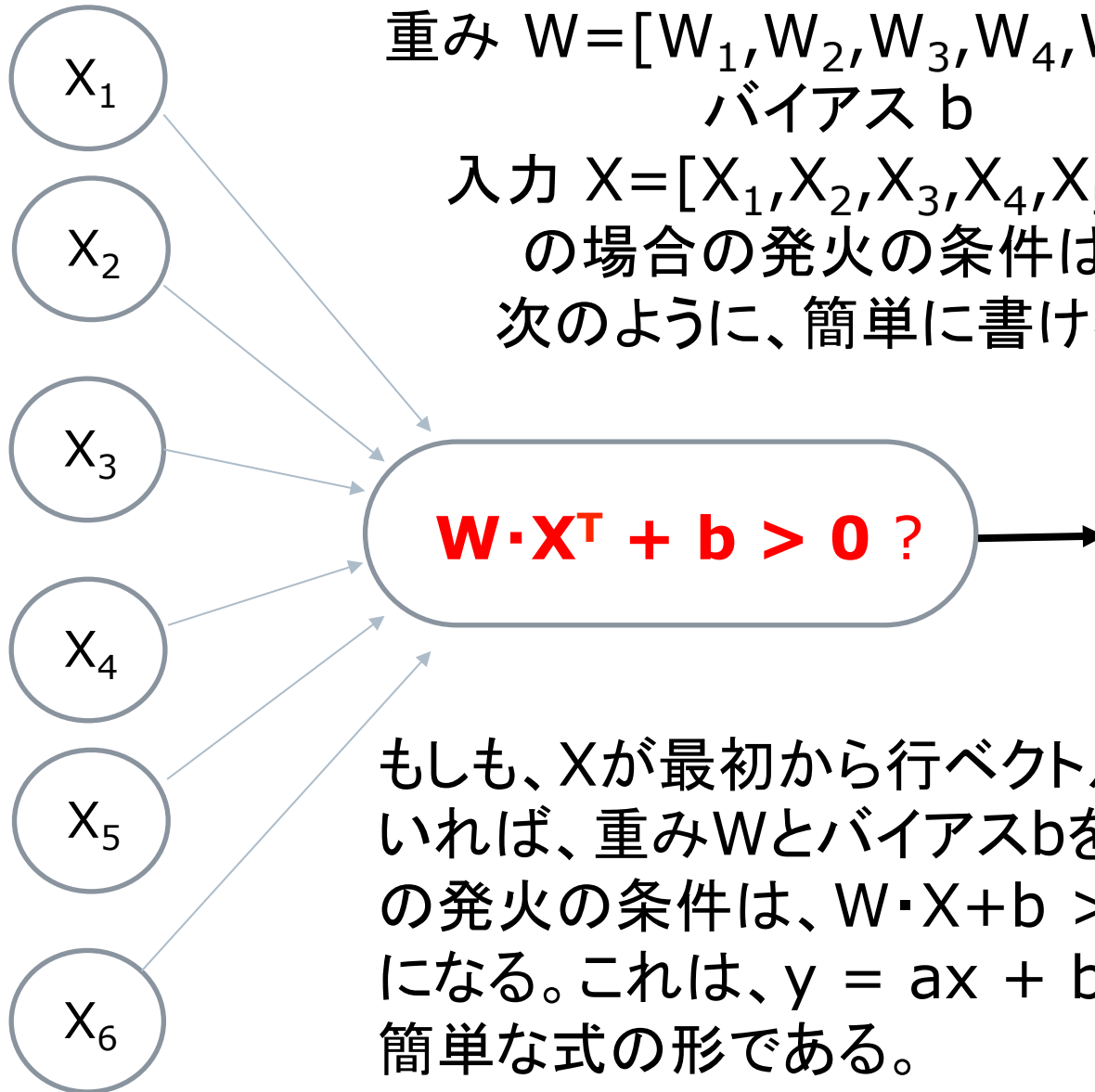
この時、

重み $W=[W_1, W_2, W_3, W_4, W_5, W_6]$

バイアス b

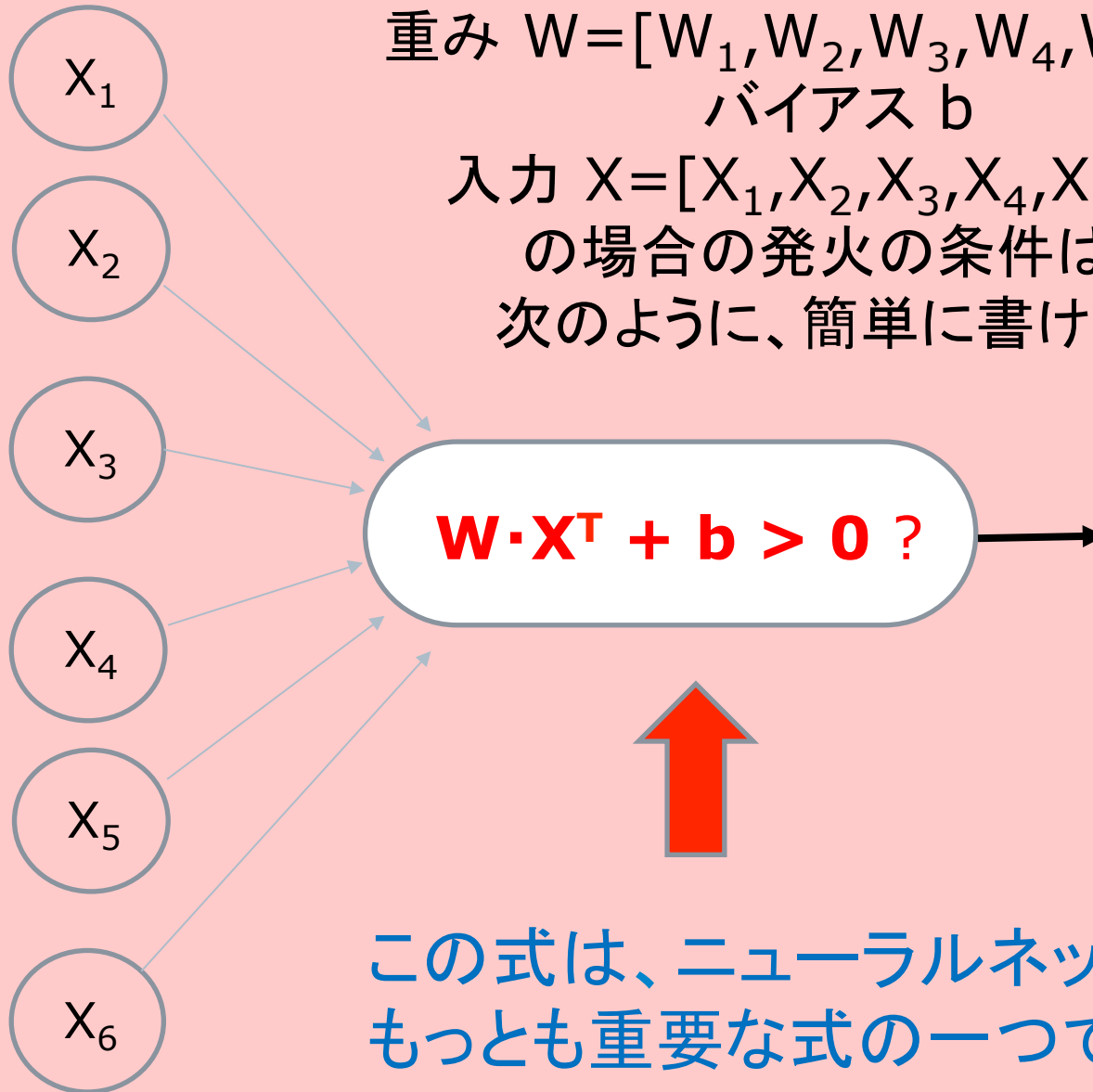
入力 $X=[X_1, X_2, X_3, X_4, X_5, X_6]$

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。



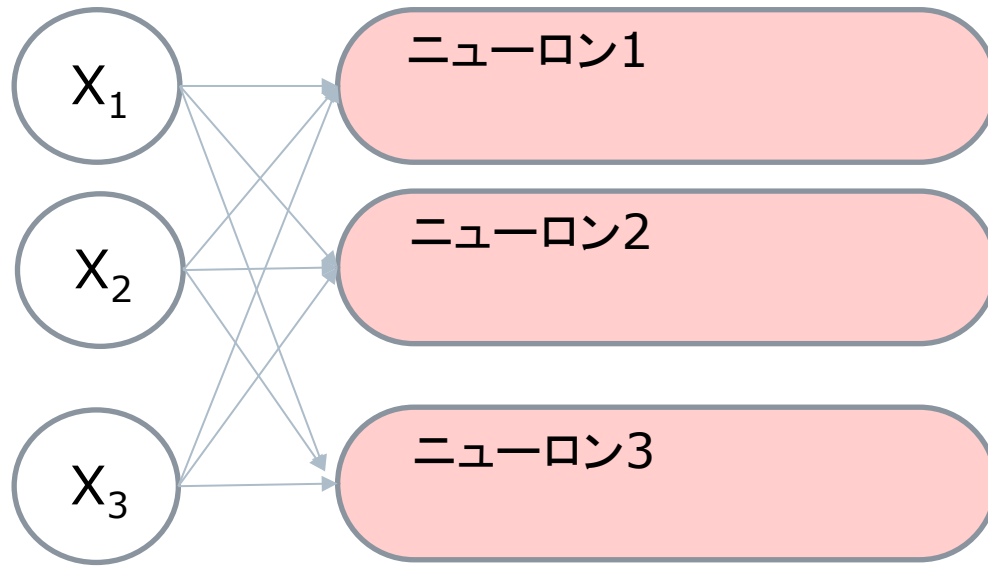
この式は、ニューラルネットワークで、
もっとも重要な式の一つである。

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。

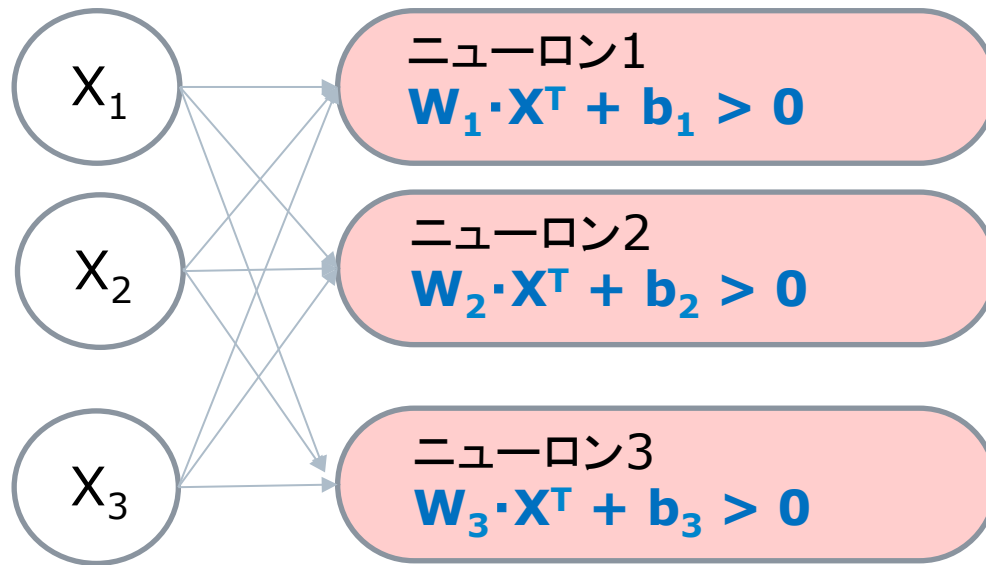
ニューロン2の重みが $[-4, 1, -5]$ バイアスが 5 なら

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

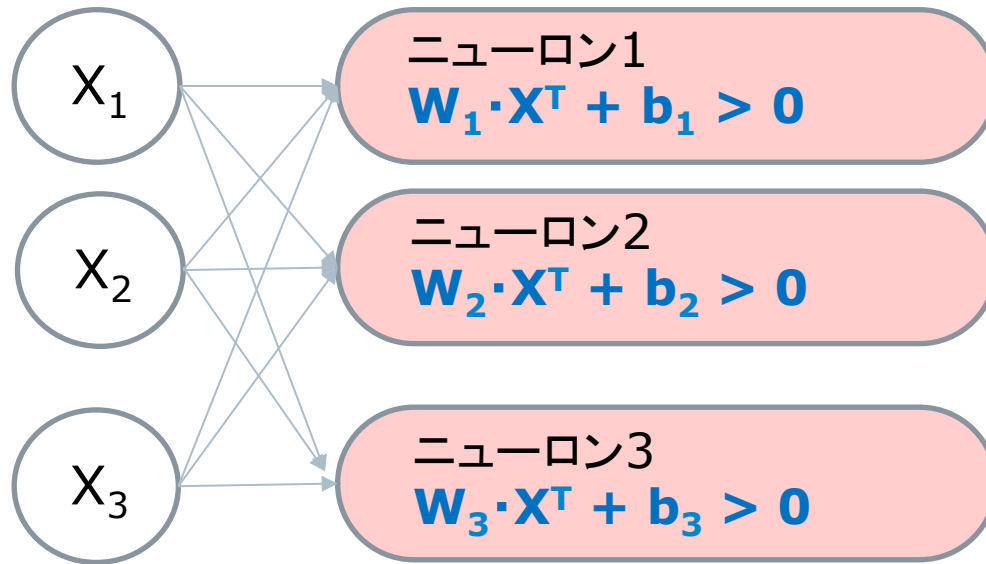
一般的には、

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

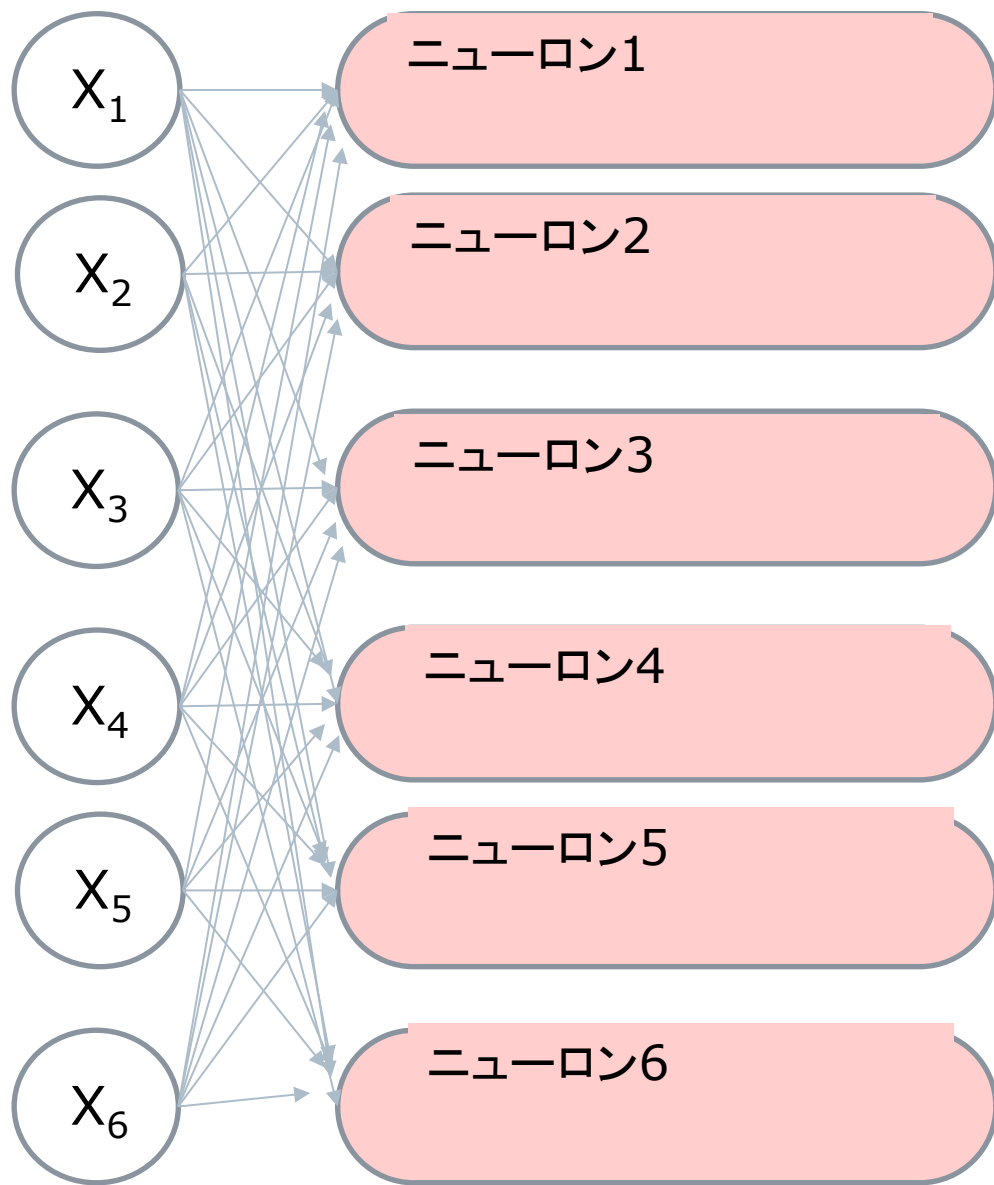
□ 複数のニューロンの層で、複数の入力の場合

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

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

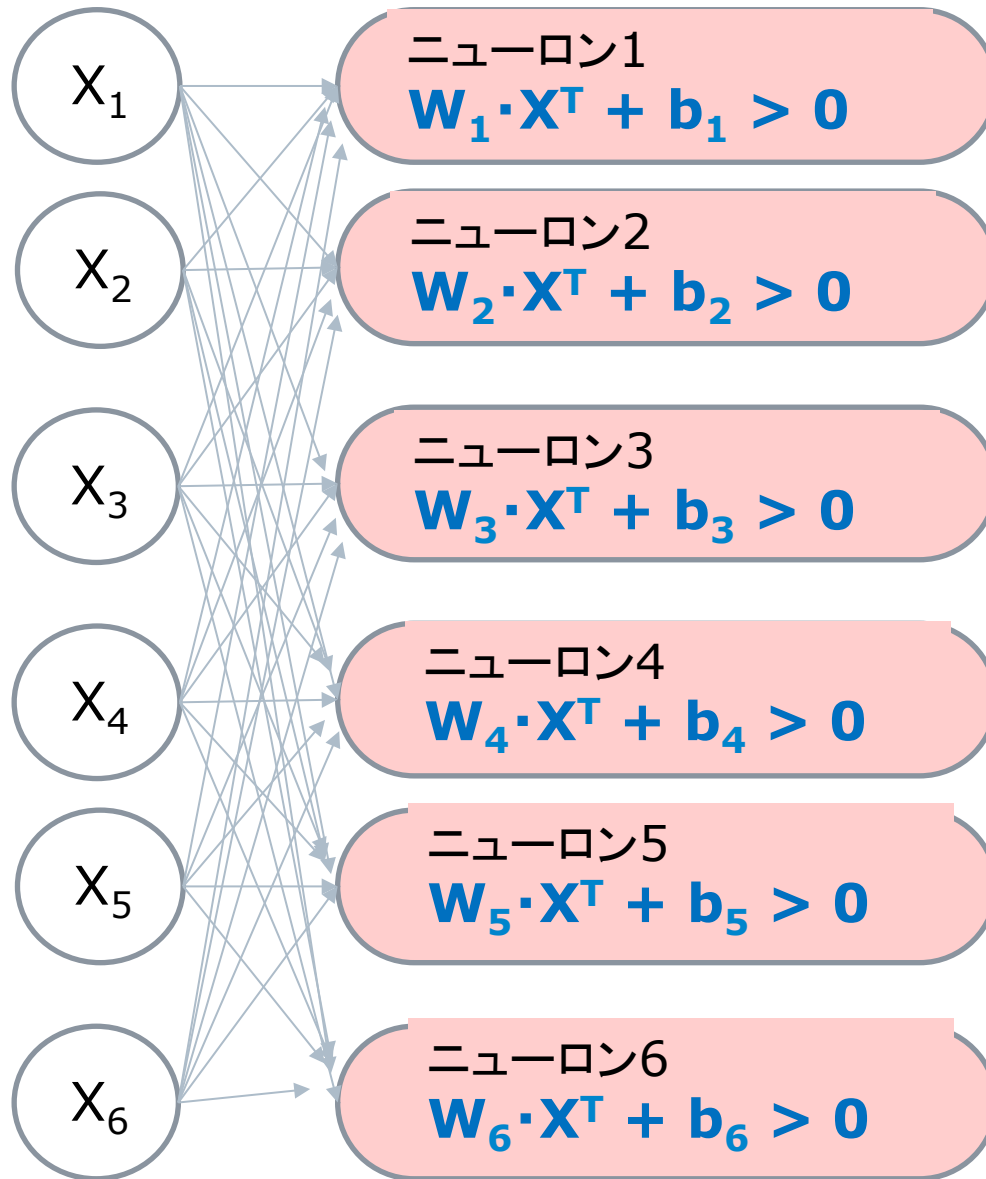
$\mathbf{W} \cdot \mathbf{X}^T$ は、行列とベクトルの積

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

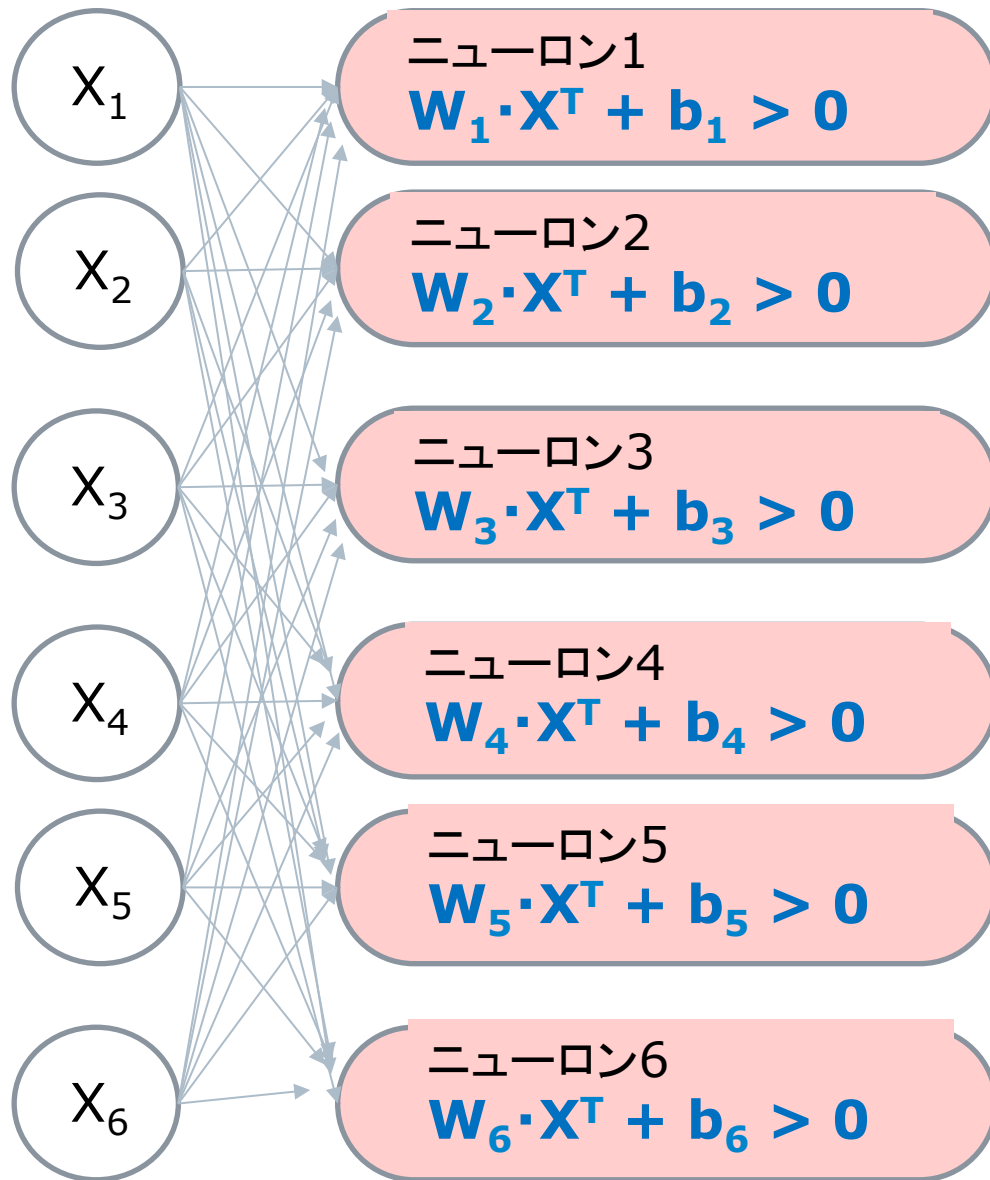
重み $W_3=[W_{31}, W_{32}, W_{33}, W_{34}, W_{35}, W_{36}]$
バイアス b_3

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

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

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

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

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

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

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。

この節は、読み飛ばしてもらって結構です。

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。
- $Y=y, X=[x_1, x_2, \cancel{x_3}]^T, W=[w_1, w_2, \cancel{w_3}], b=b_0$ なら、
 $y = w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + b_0$
- $Y=y, X=[x_1, x_2, \cancel{x_3}, x_4]^T, W=[w_1, w_2, \cancel{w_3}, w_4], b=b_0$ 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。 $y_1 = 2x_1 + 3x_2 + 1$, $y_2 = 3x_1 + 5x_2 + 2$
- $Y=[y_1, y_2, y_3]^T$, $X=[x_1, x_2, x_3]^T$, $W = \begin{bmatrix} [2,3,-1], [4,-5,1], [1,2,3] \end{bmatrix}$, $b=[1,2,3]^T$ 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。

ニューラル・ネットワークとその表現(2)

- ニューロンを動かす -- 活性化関数 (Activator)
- クラス分けのActivator -- SoftMax

ニューロンを動かす

-- 活性化関数 (Activator)

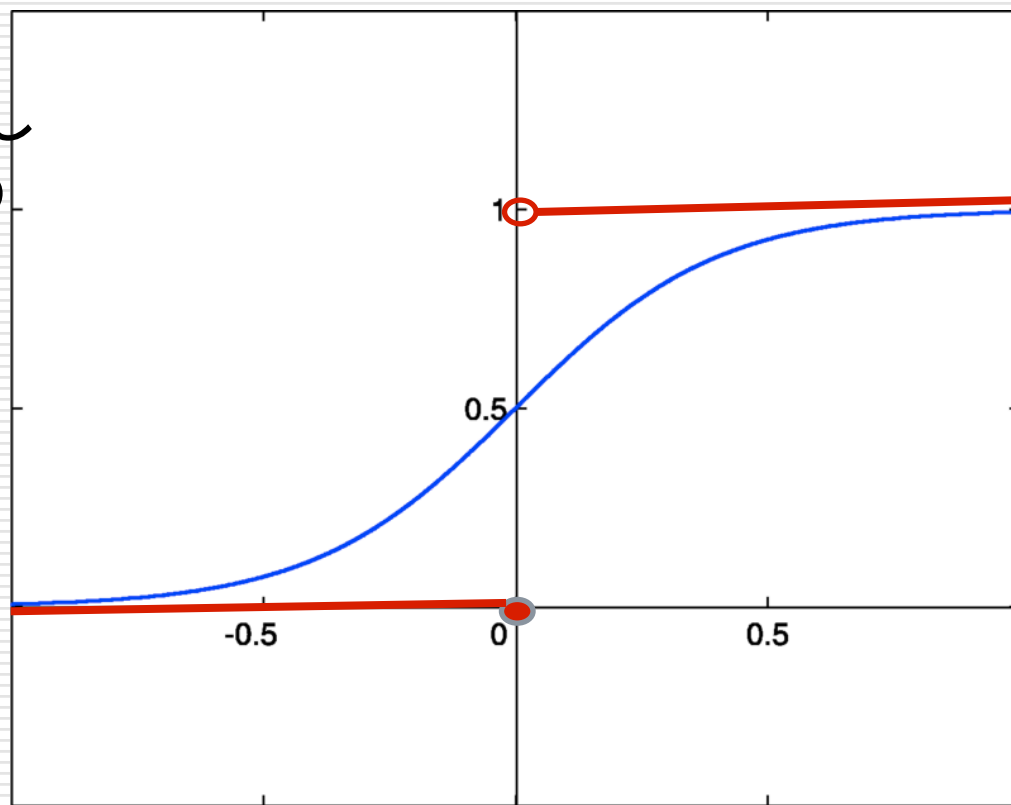
活性化関数は、その名前の通りに、一つのニューロンが発火(活性化)する時の条件と出力を関数の形で表したものである。いろいろな種類の活性化関数がある。

活性化関数

- 先に見たように、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。
$$\phi(x) = \begin{cases} 0 & x \leq 0 \text{ の時} \\ 1 & x > 0 \text{ の時} \end{cases}$$

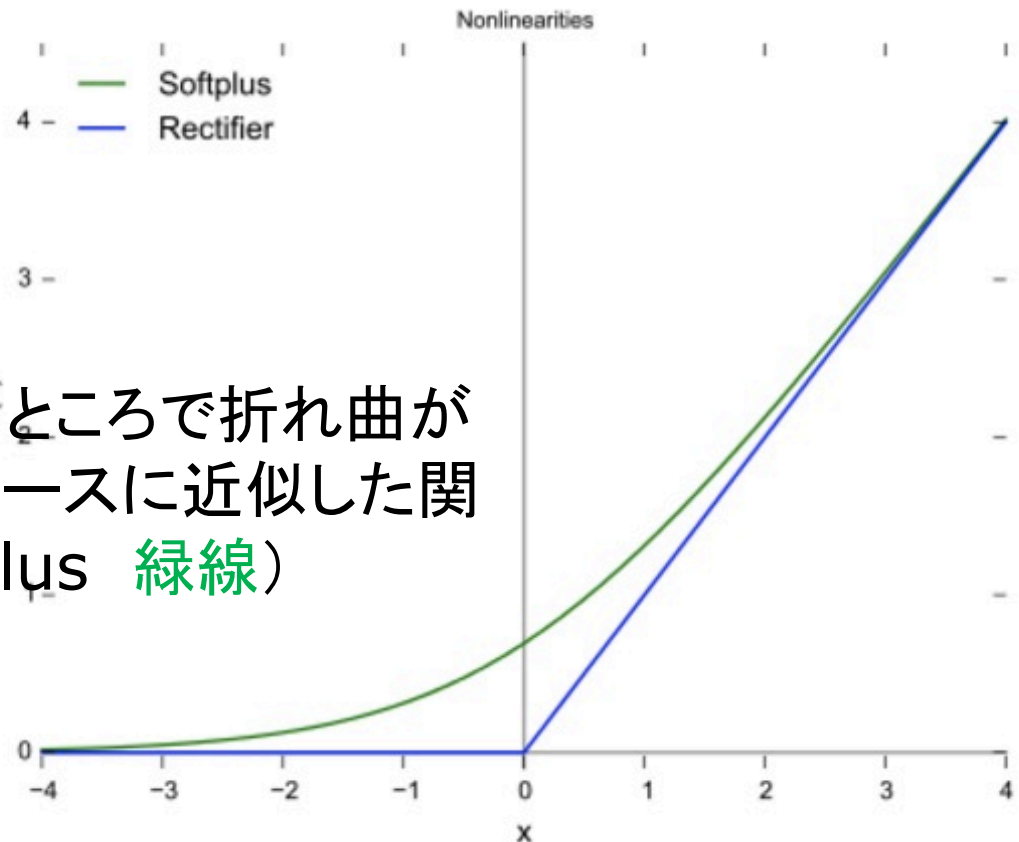
sigmoid 関数 (logistic 関数)

- 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。



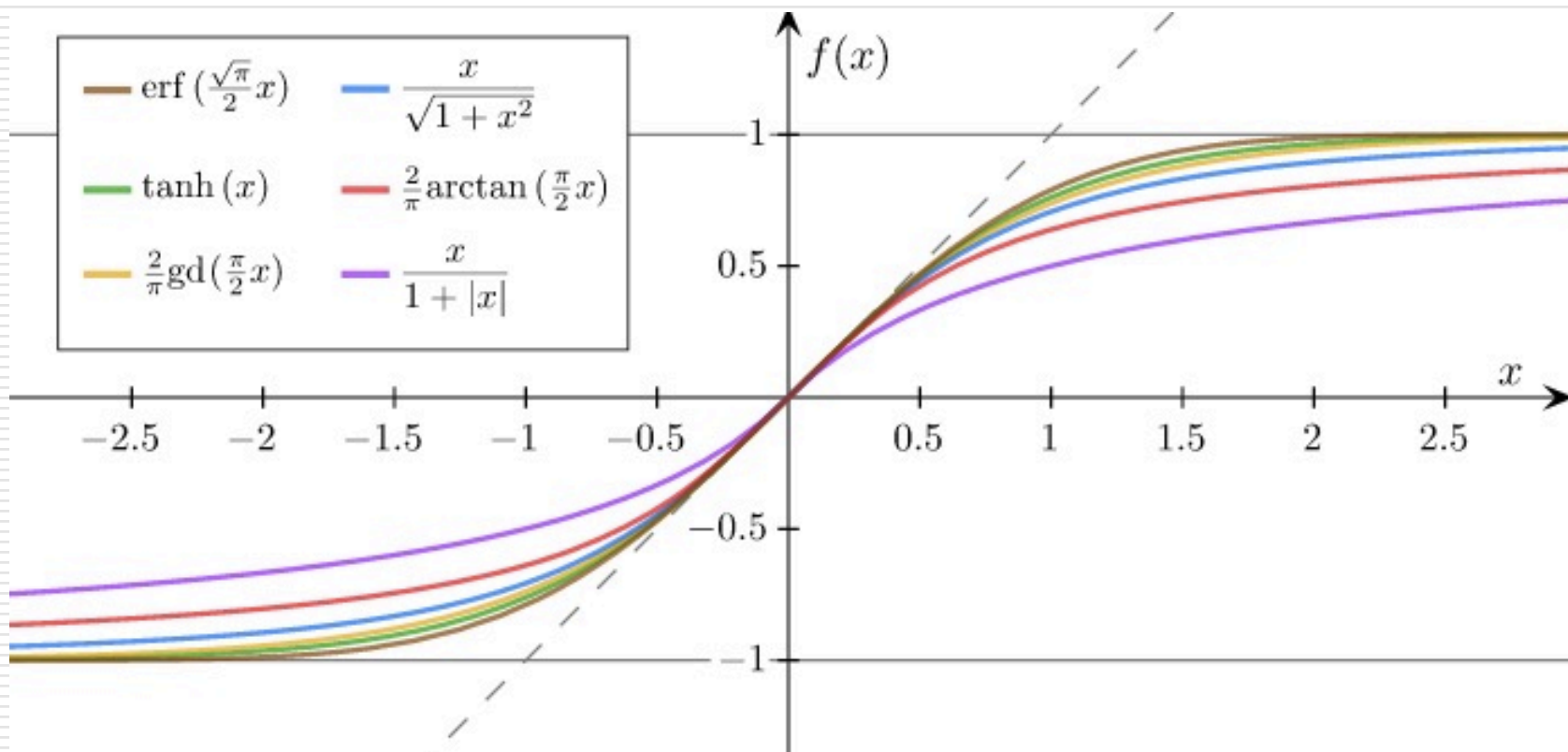
ReLU (rectified linear unit)

- ❑ 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。



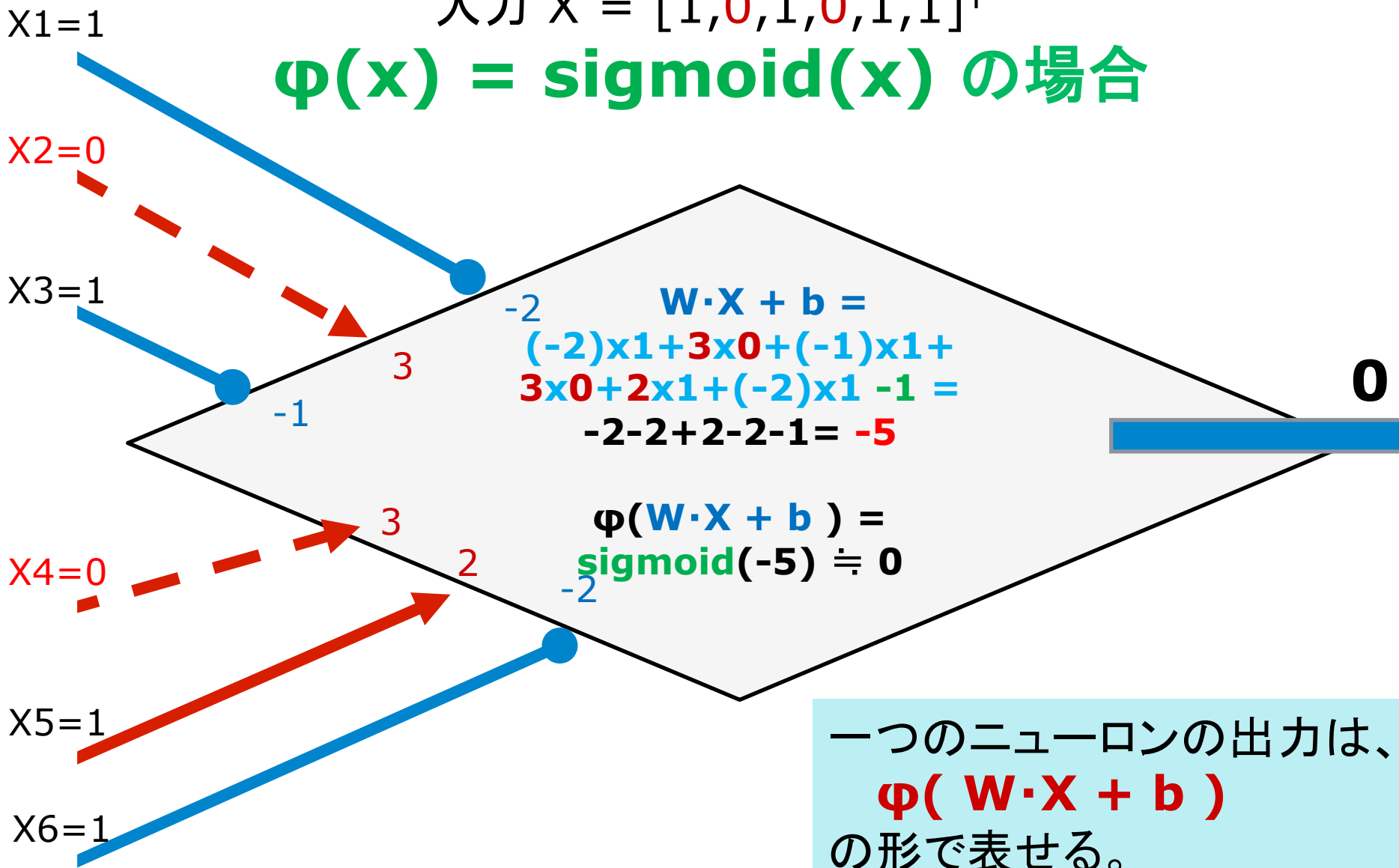
tanh 関数

- 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 $X = [1, 0, 1, 0, 1, 1]^T$

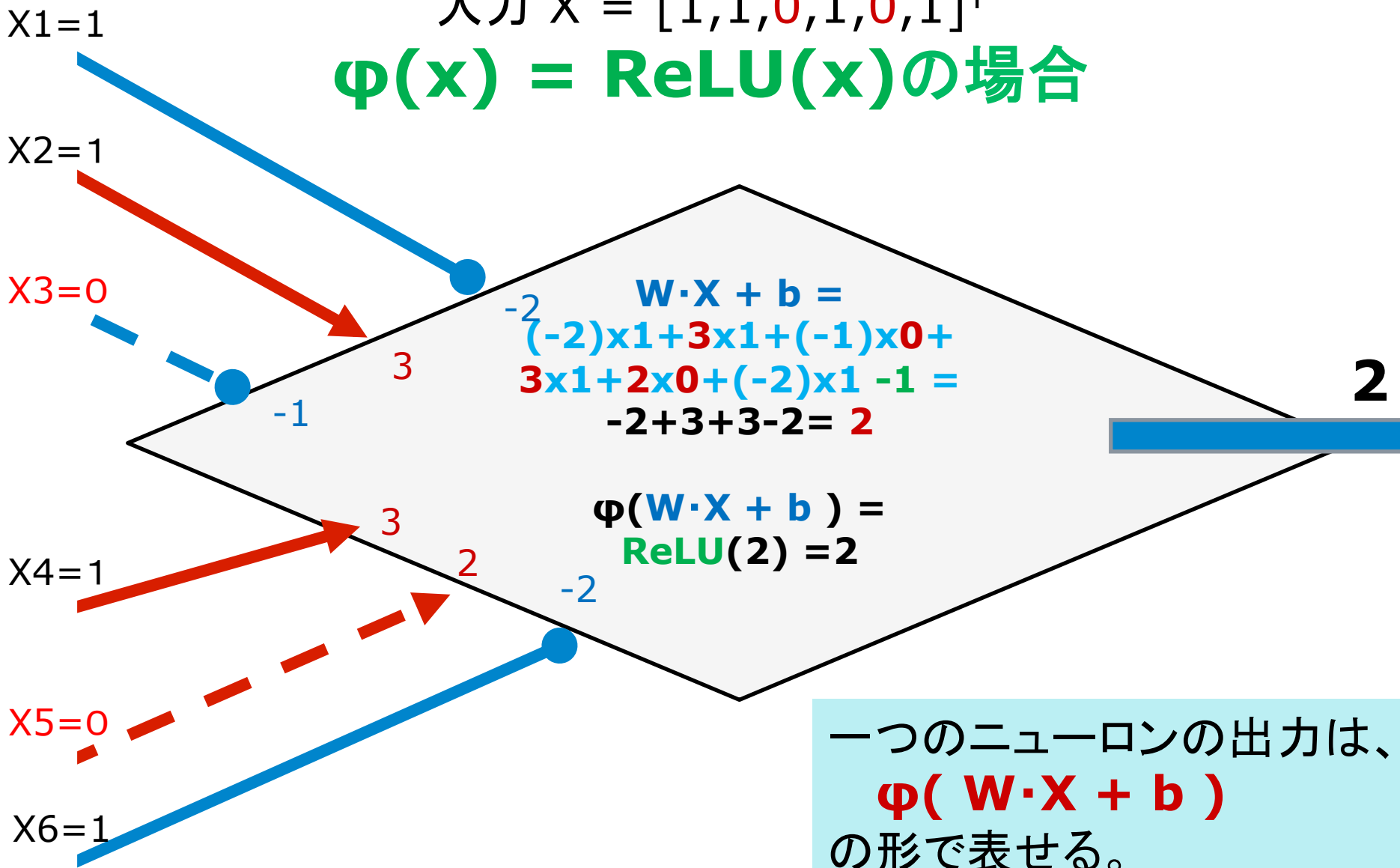
$\phi(x) = \text{sigmoid}(x)$ の場合



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



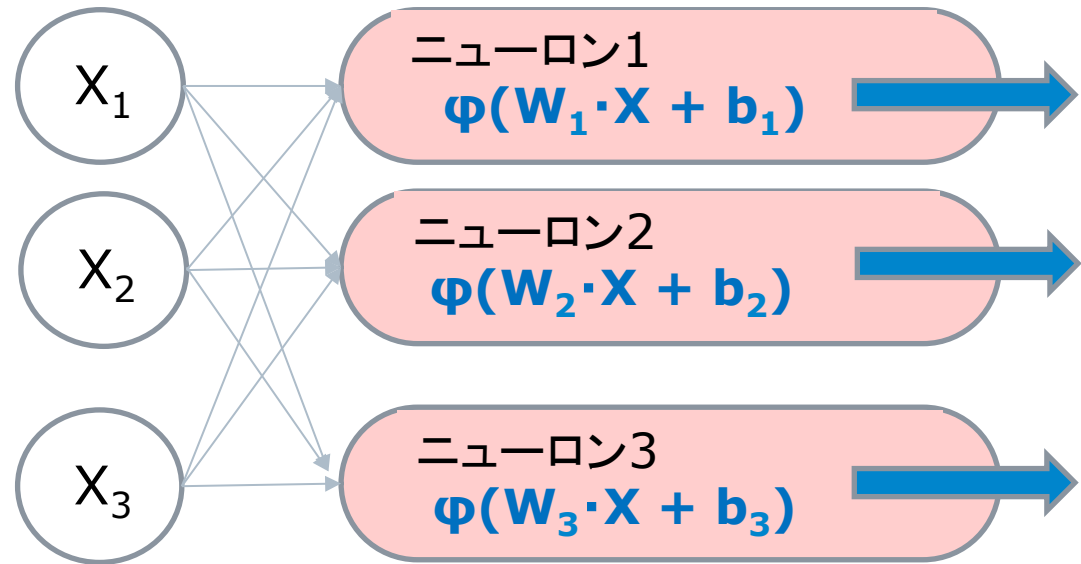
一つのニューロンの出力は、
 $\phi(W \cdot X + b)$
の形で表せる。

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。

クラス分けのActivator SoftMax

画像認識等のクラス分けの出力には、Softmaxという活性化関数がよく利用される

softmax 関数

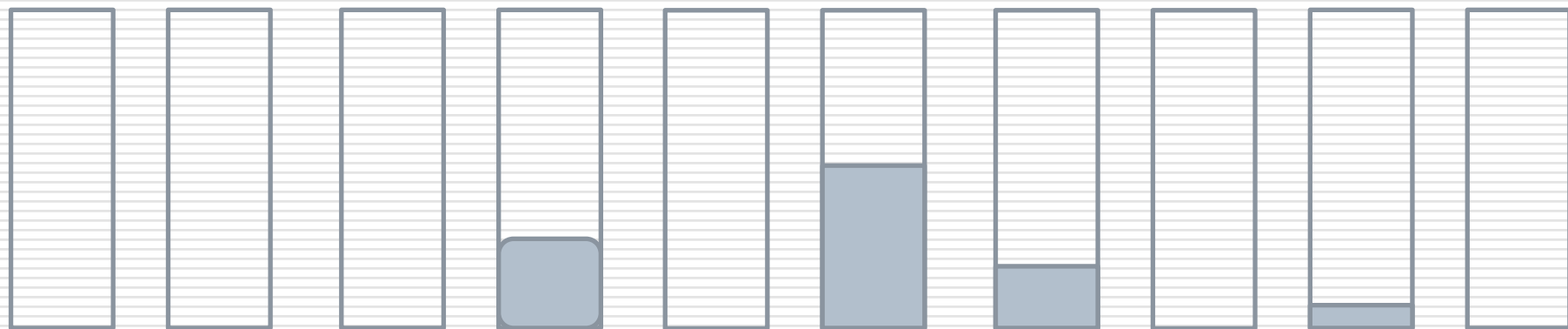
- クラス分けの場合の出力に利用されるactivator。
手書き数字画像の認識だと、提示された画像が0から9までの10個の数字のどれかだと判断しなければいけない。
 - ニューラル・ネットワークでは、こうした場合、出力層に、0から9までの10個の数字に対応した10個のノードが並び、それぞれのノードの値が、「提示された画像がその数字である確率」であるようにする。
 - ノードごとに、確率を計算するのが softmax関数である。最終的には、10個のノードの中で、その確率が最大であるものが選ばれる。
-

softmax 関数の出力例



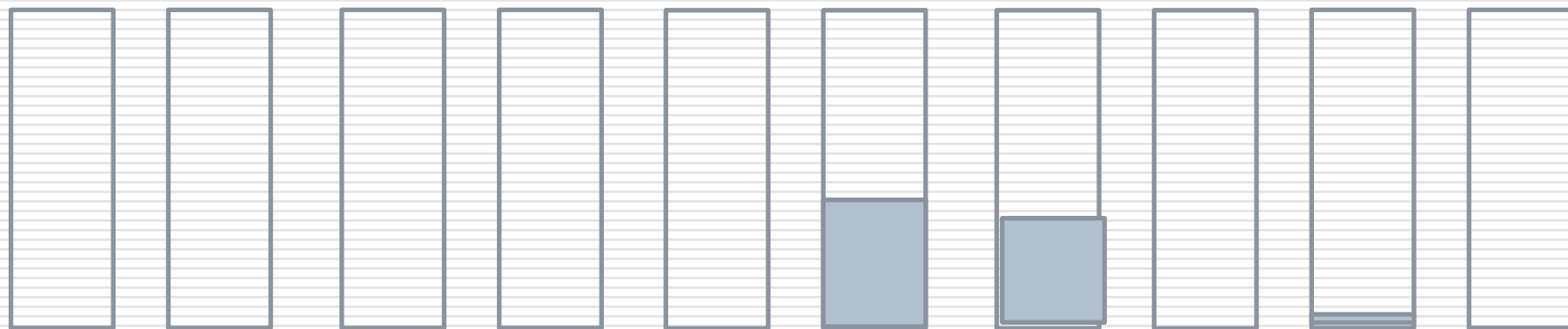
ノード0 ノード1 ノード2 ノード3 ノード4 ノード5 ノード6 ノード7 ノード8 ノード9

softmax 関数の出力例



ノード0 ノード1 ノード2 ノード3 ノード4 ノード5 ノード6 ノード7 ノード8 ノード9

softmax 関数の出力例



ノード0 ノード1 ノード2 ノード3 ノード4 ノード5 ノード6 ノード7 ノード8 ノード9

softmax 関数の性質

- softmax関数は、それが適用される層のノードの softmax関数の値をすべて足し合わせると1になるという性質を持つ。それは、softmaxの出力が「確率」であることを考えれば、すぐわかる。
- 例えば、先の例で、ある画像が3である確率が 0.75であったとしよう。そのことは、その画像が3以外の数字である確率は、 $1 - 0.75 = 0.25$ だということである。3以外のノードの確率(softmaxの出力の値)を全部足しても、0.25にしかないということである。
- 次のこともわかる。あるノードのsoftmaxの値が上がれば、他のノードの値は下がる。あるノードの値を下げれば、他のノードの値は上がる。全部足して1になるように、値は変化する。

One-Hot-Vector

- 先のsoftmax関数の出力例の、一番上のノードは、「この画像は数字nである」という、ニューラル・ネットワークの判断を表しているの。もちろん、softmaxの値が一番大きいところ選ばれて、そこに一つだけに1が入っていて、残りはすべて0になっている。
- こうした一つだけに1が、残りすべてが0のベクトルを、One-Hot-Vector と呼ぶ。
- One-Hot-Vectorは、クラス分けの出力に現れるだけではない。ニューラル・ネットワークを「学習」させるための訓練用データは、画像と一緒に「この画像は数字nである」という「正解」の情報を持っていなければならない。訓練用データは、画像と正解を示すOne-Hot-Vectorの二種類のデータから構成されている。



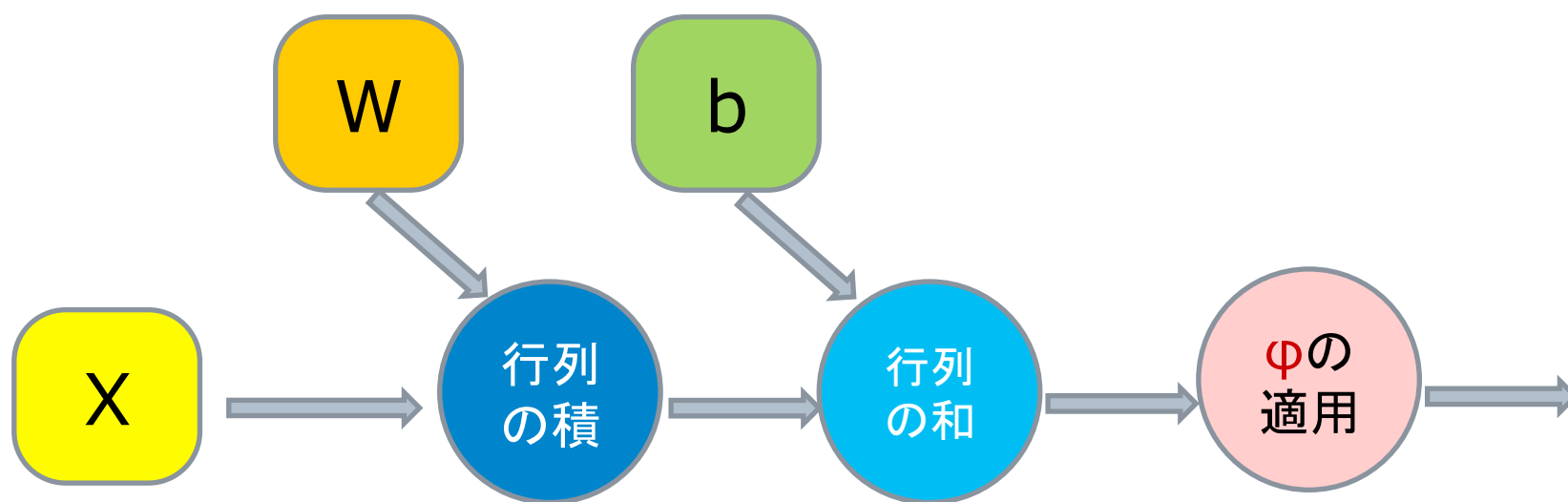
ニューラル・ネットワークを
グラフで表現する

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。

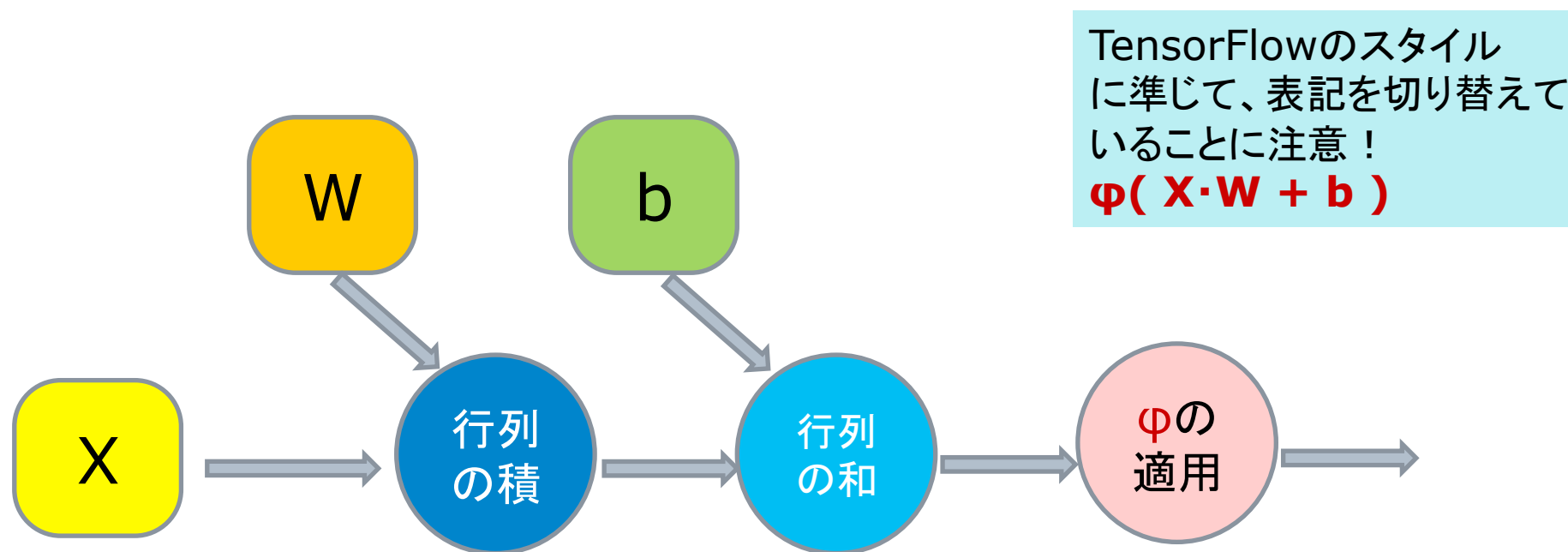
「グラフ」「演算ノード」「テンソル」というのが、基本的なコンセプトとなる。

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。



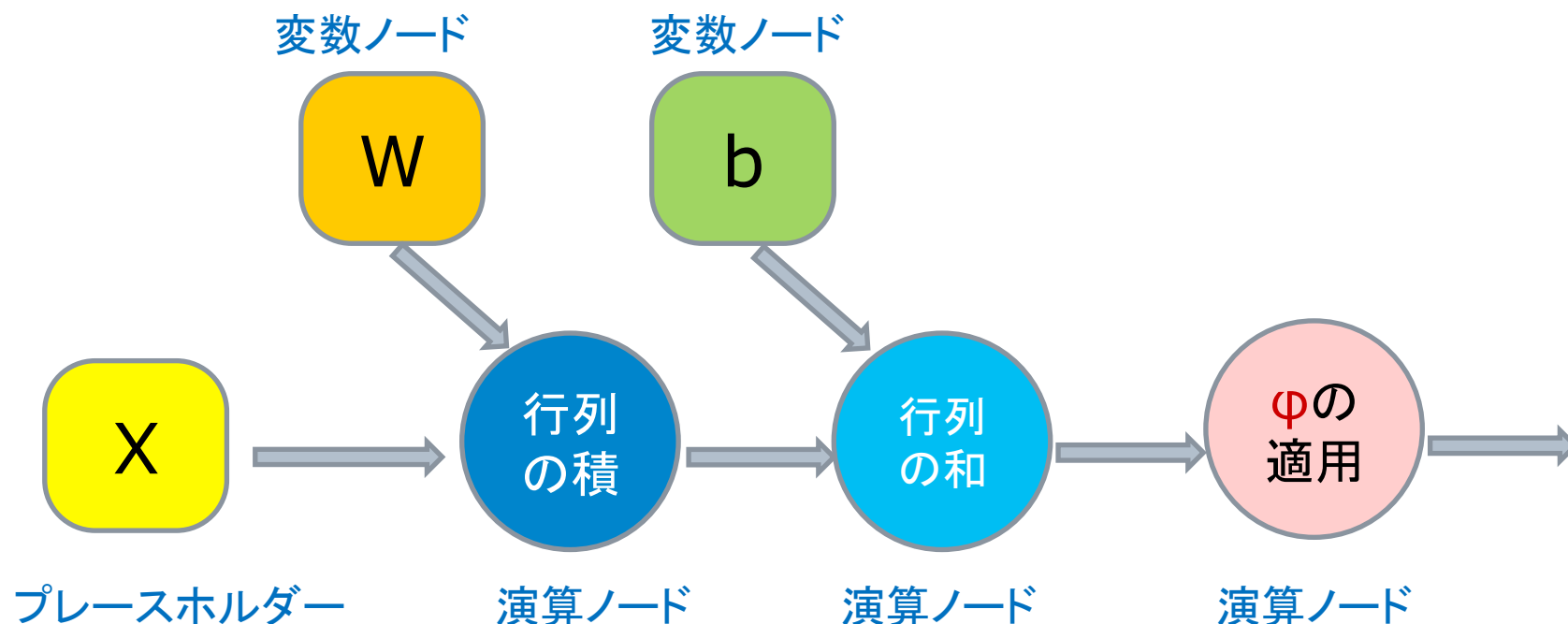
先の例で言うと、
 X は、 $[X_1, X_2, X_3]^T$ の列ベクトルで、
 W は 3×3 の行列、
 b は、 $[b_1, b_2, b_3]^T$ の列ベクトルである。

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。



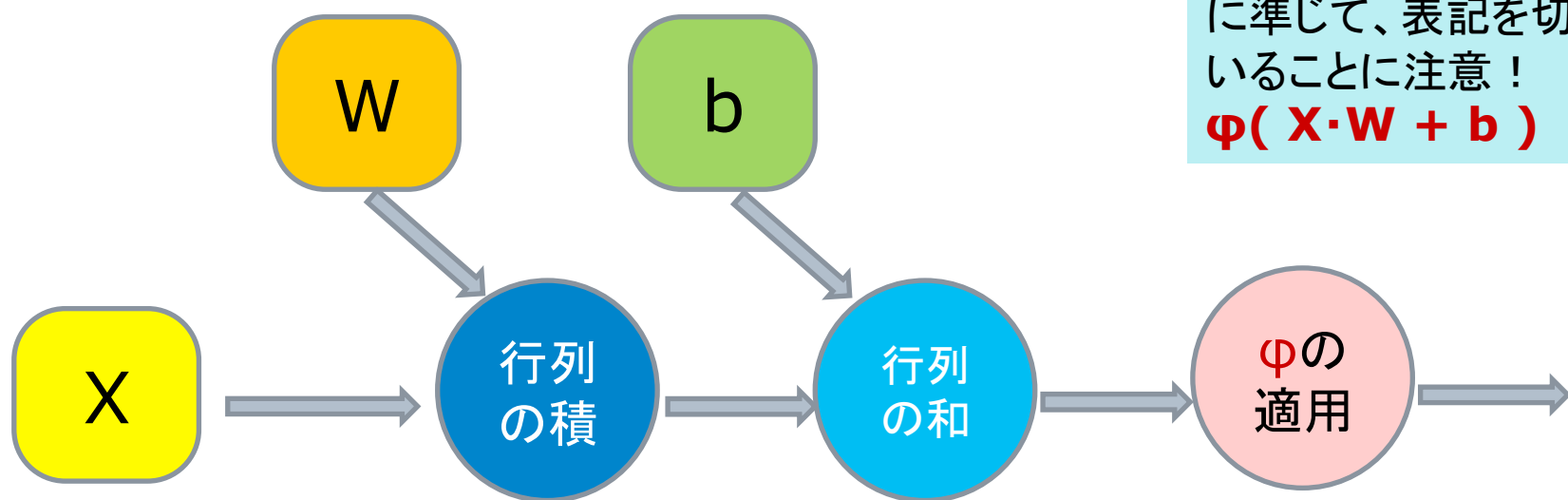
先の例で言うと、
Xは、 $[X_1, X_2, X_3]$ の行ベクトルで、
Wは 3×3 の行列、
bは、 $[b_1, b_2, b_3]$ の行ベクトルである。

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)

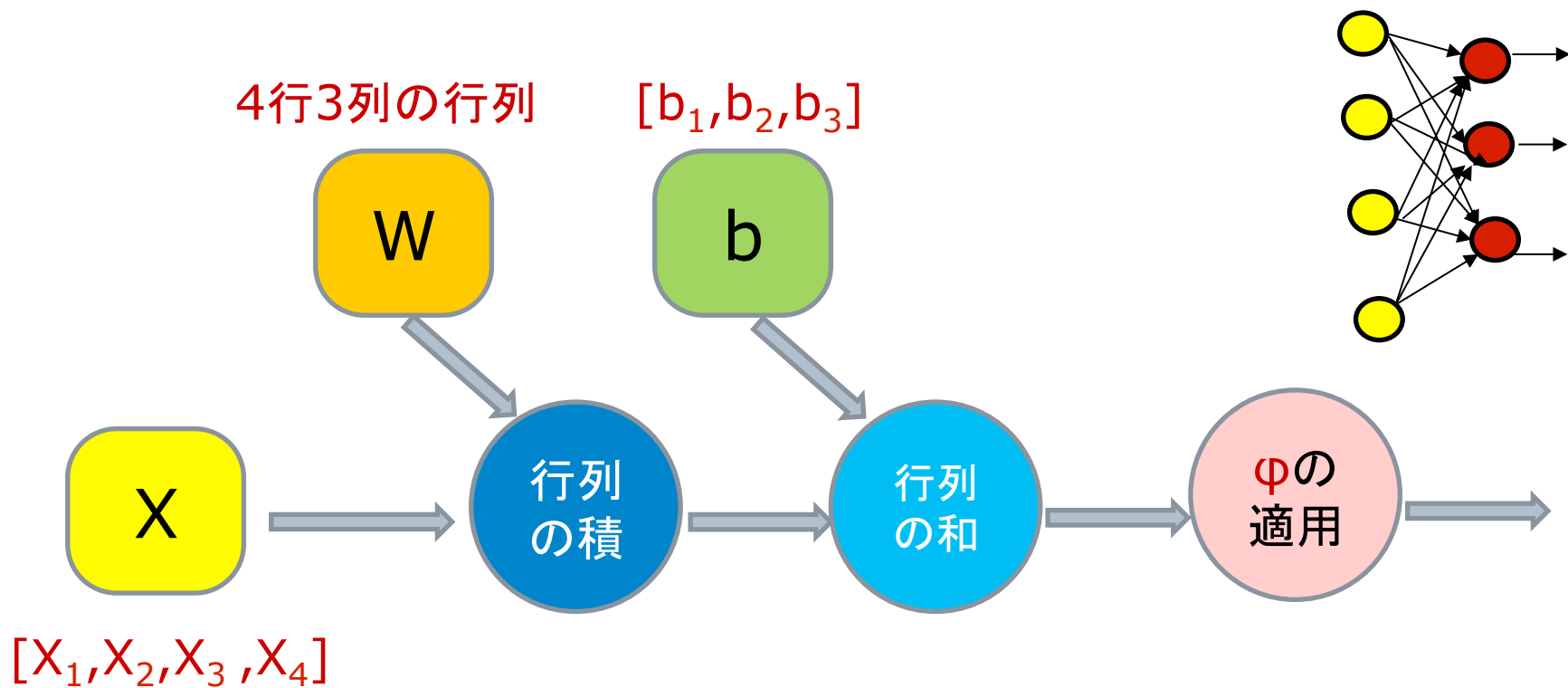


TensorFlowのスタイル
に準じて、表記を切り替えて
いることに注意！
 $\phi(X \cdot W + b)$

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。

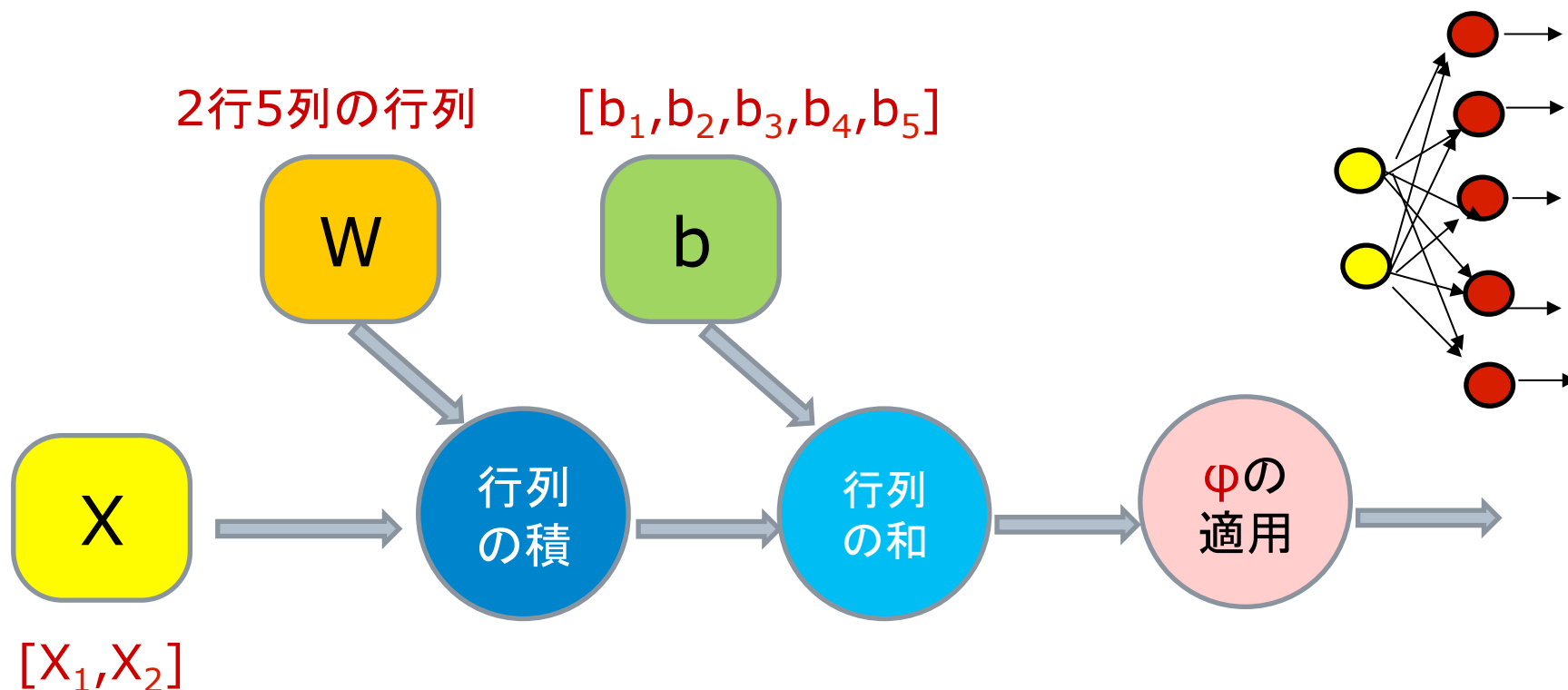
いくつかの例で考えてみよう。

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。



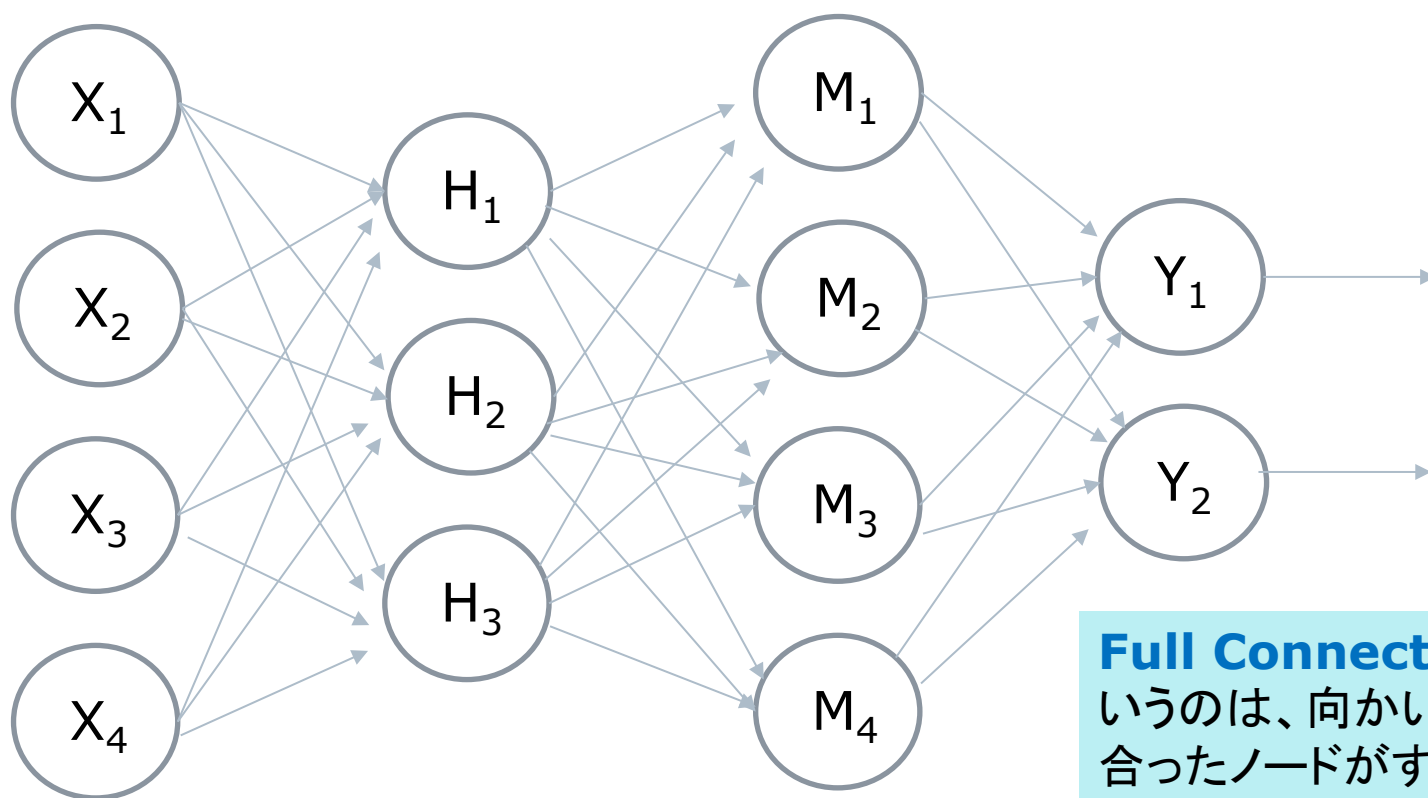
一般に、 n 個の入力を受け取り、 m 個の出力を返すニューロンの「層」では、 X は n 次元の行ベクトル、 W は n 行 m 列の行列、 b は m 次元の行ベクトルである。

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。

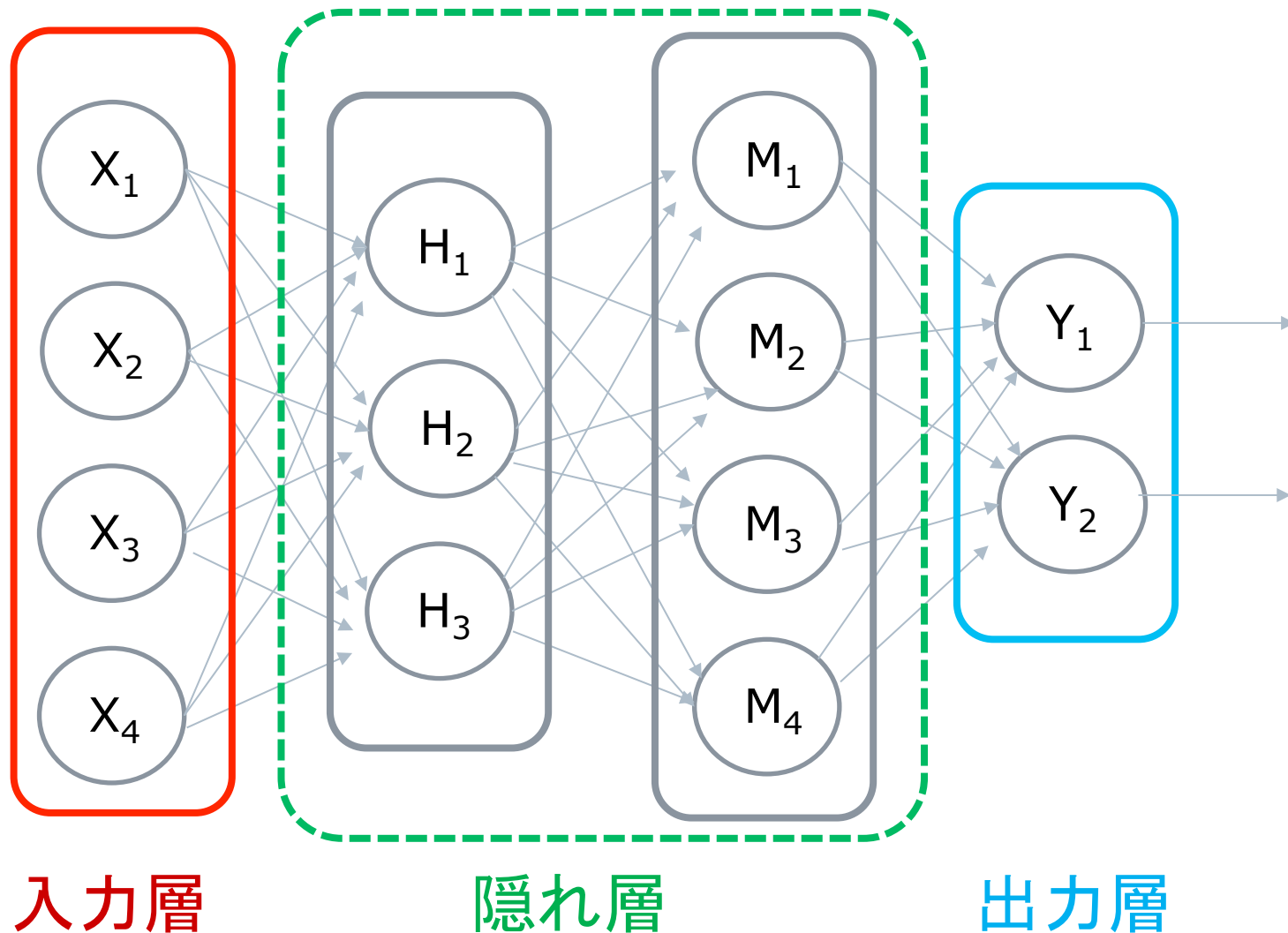
Full Connectなニューラル・ネットワークの 旧来のグラフでの表現



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

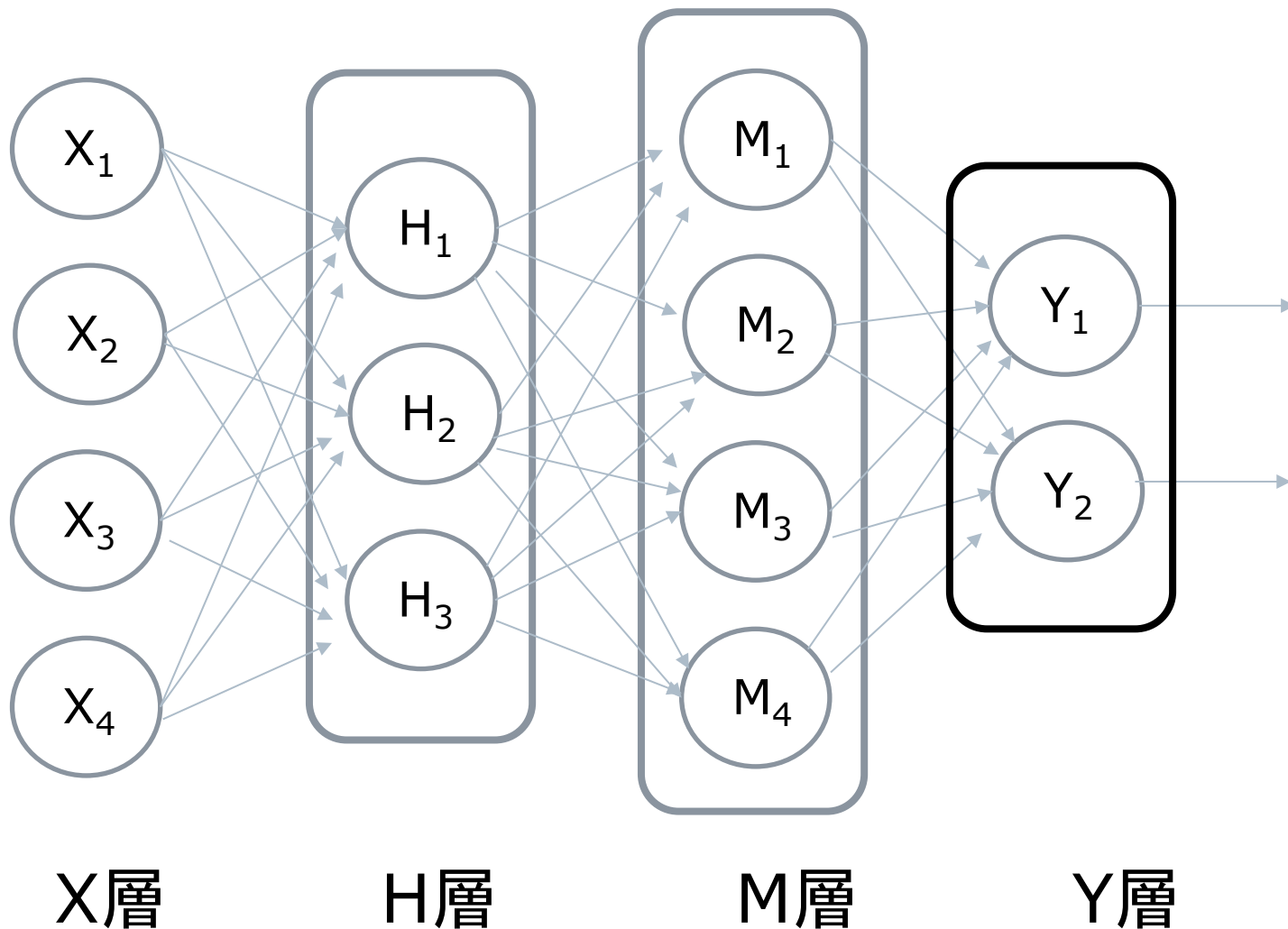


旧来のグラフ

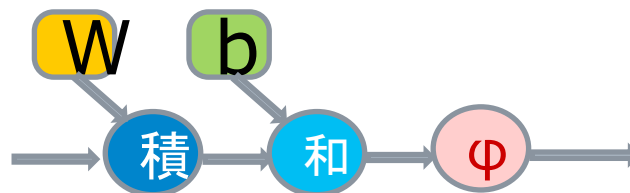
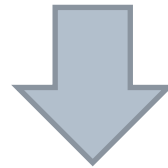
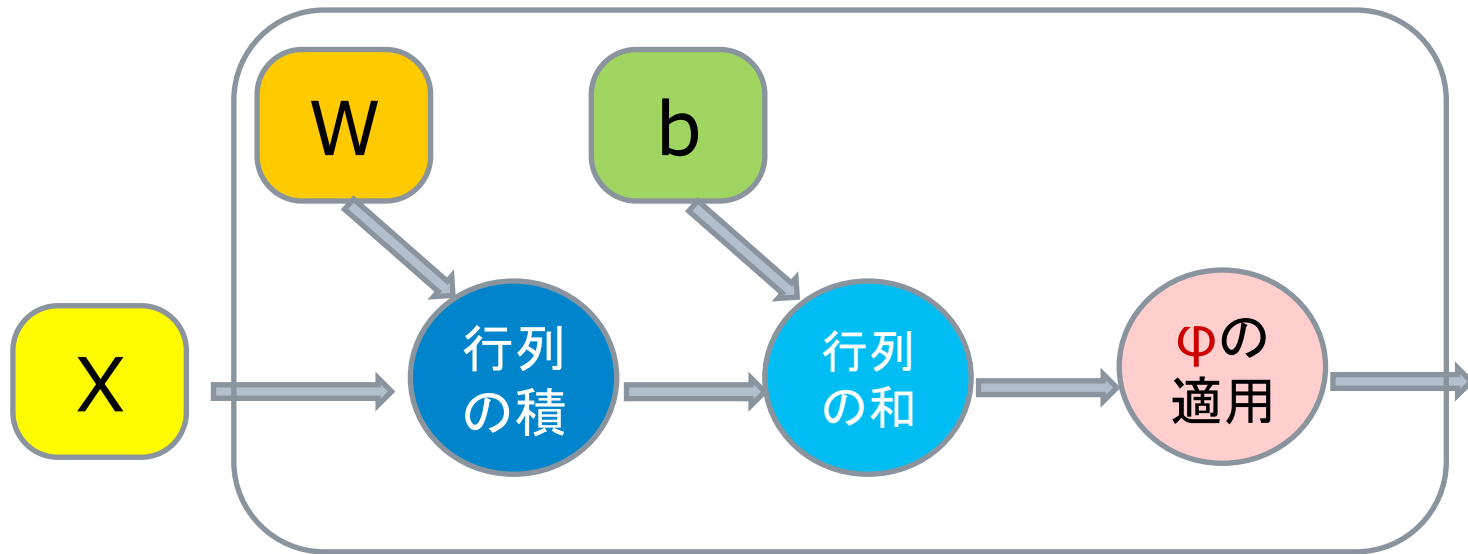
H層の
重み W_H
バイアス b_H
活性化 ϕ_H

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

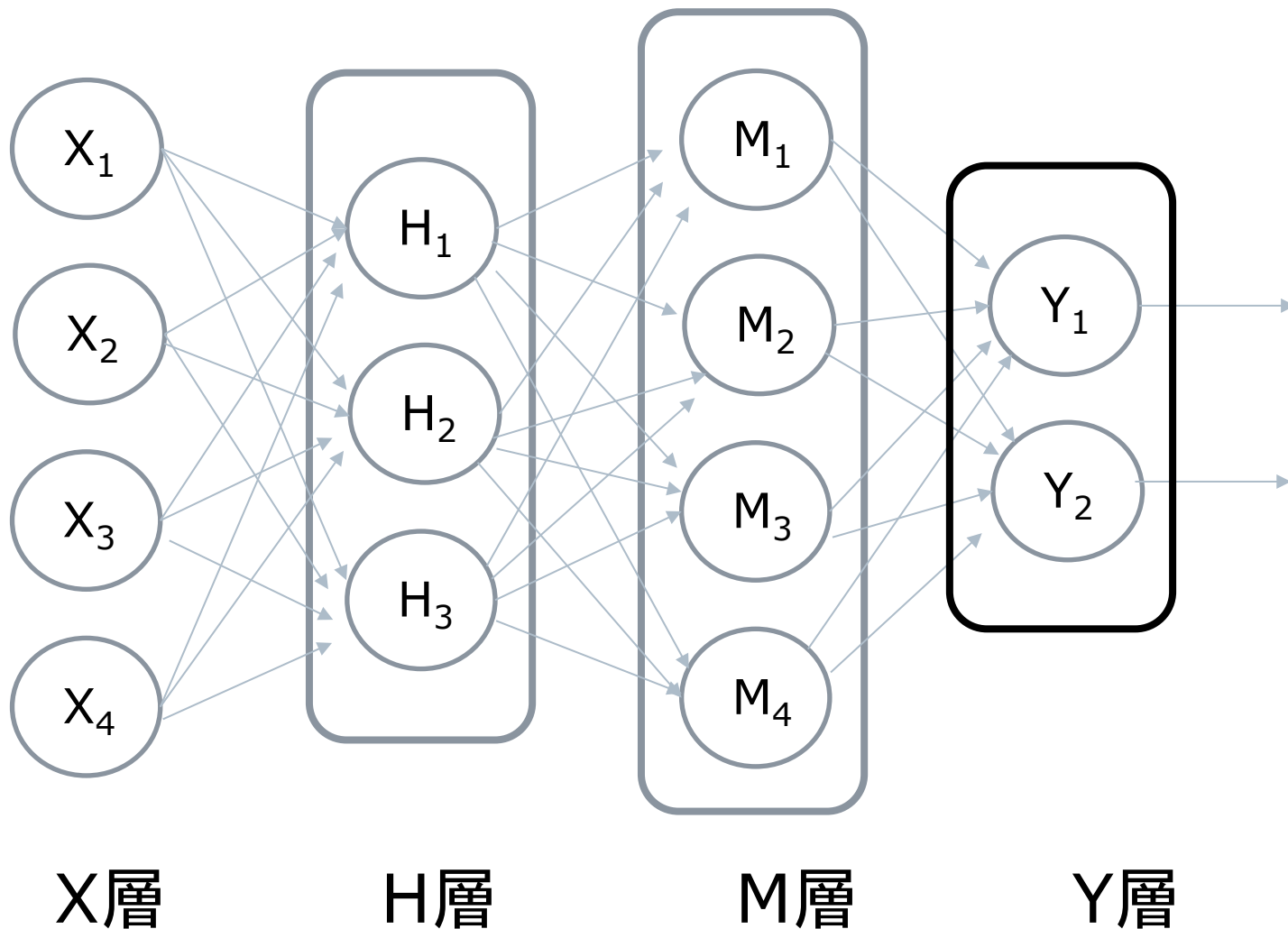


旧来のグラフ

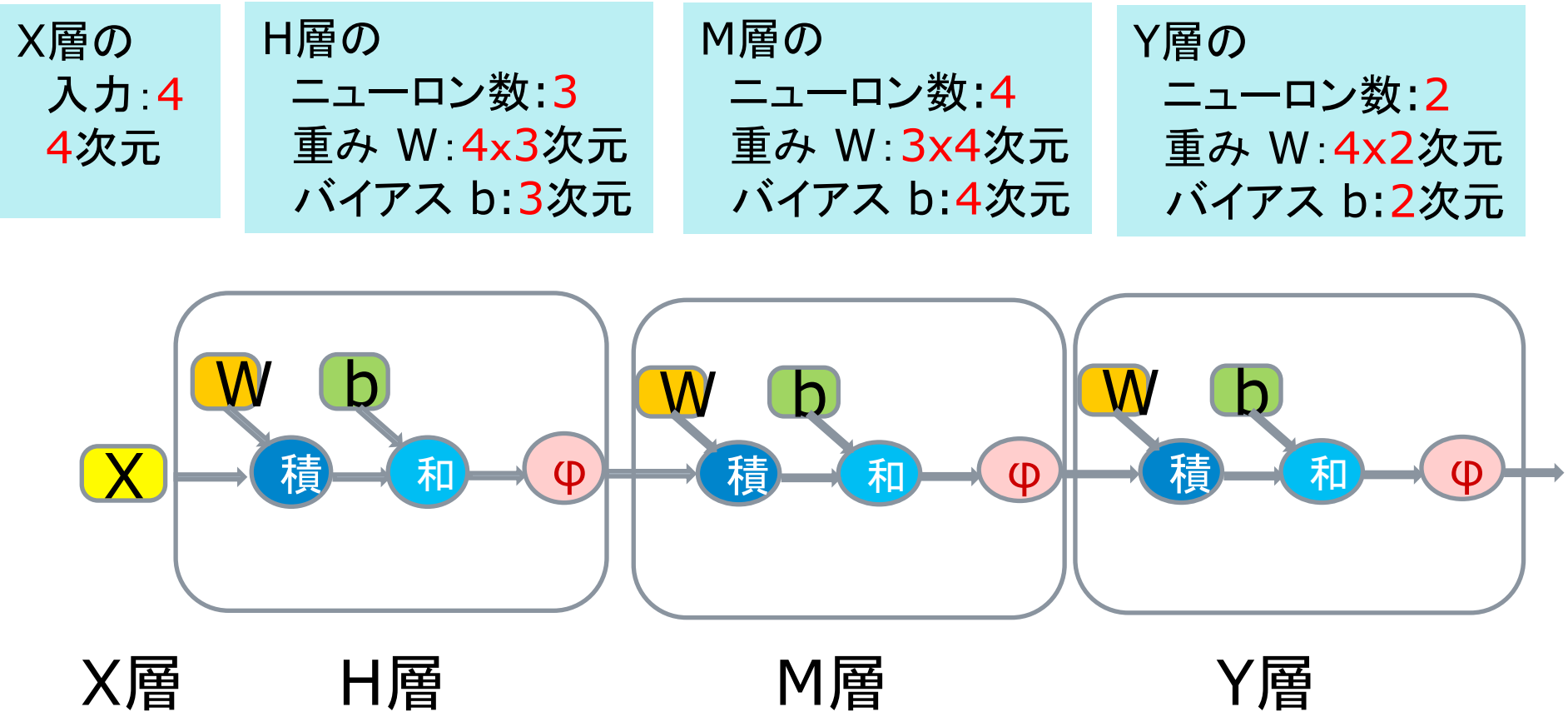
H層の
重み W_H
バイアス b_H
活性化 ϕ_H

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



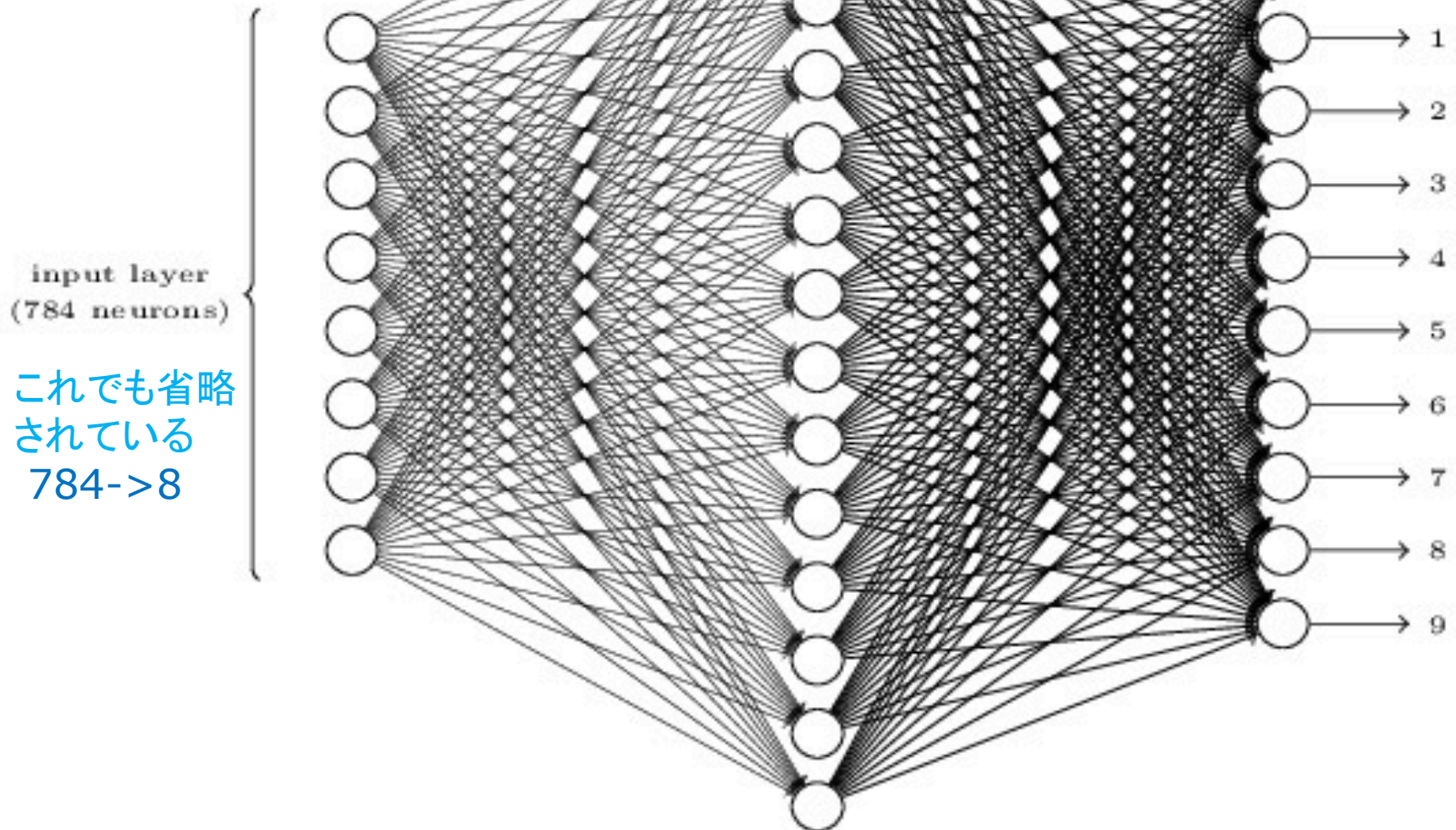
TensorFlowのグラフ



こっちのグラフの方が、ずっとわかりやすい！

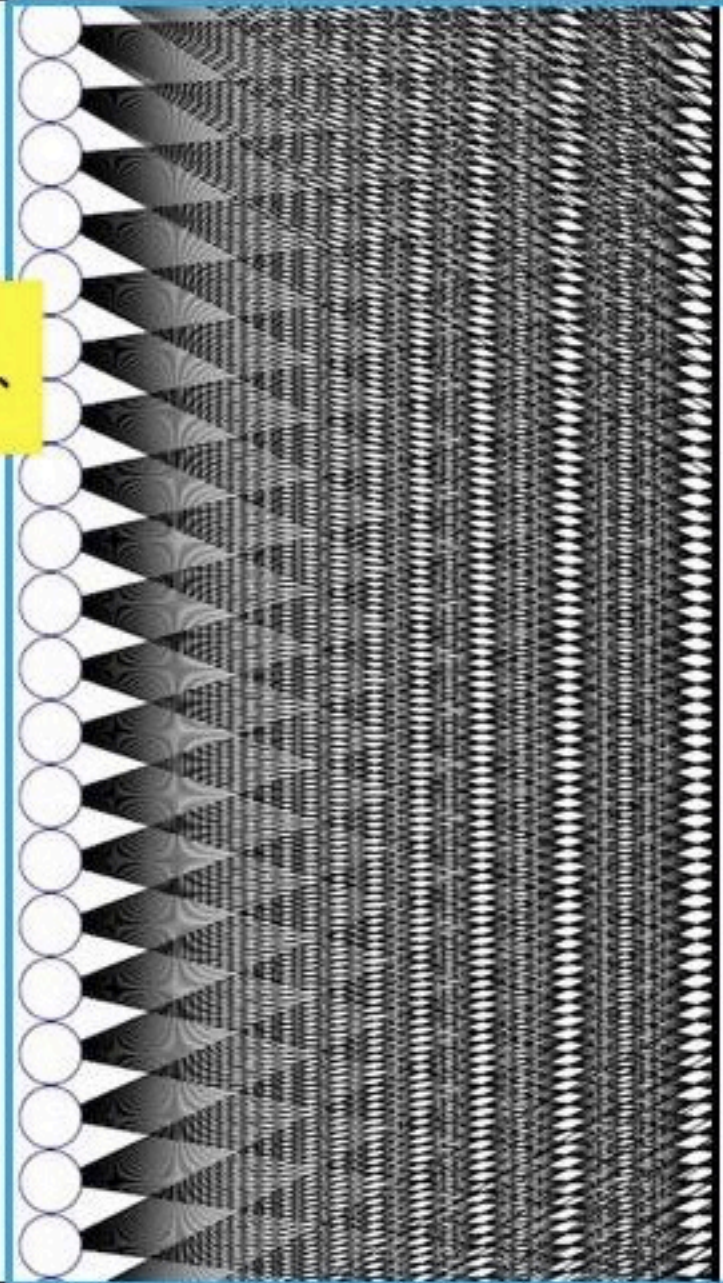
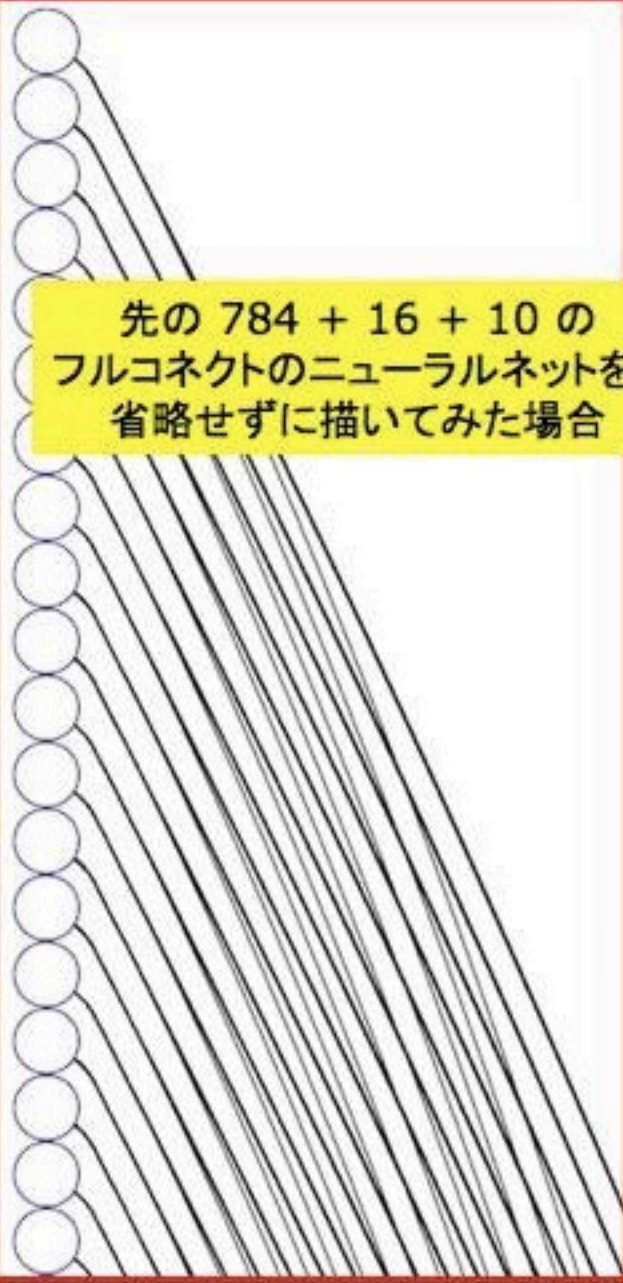
ニューラル・ネットワークの例

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

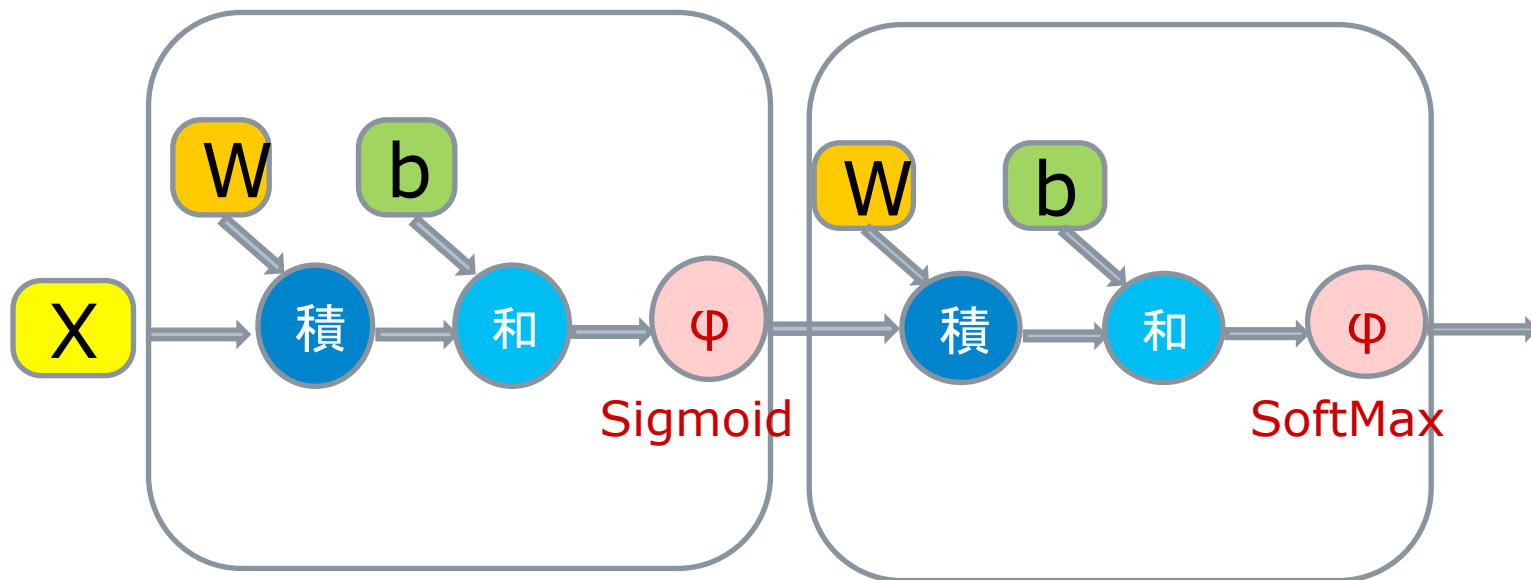


TensorFlowのグラフ

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)

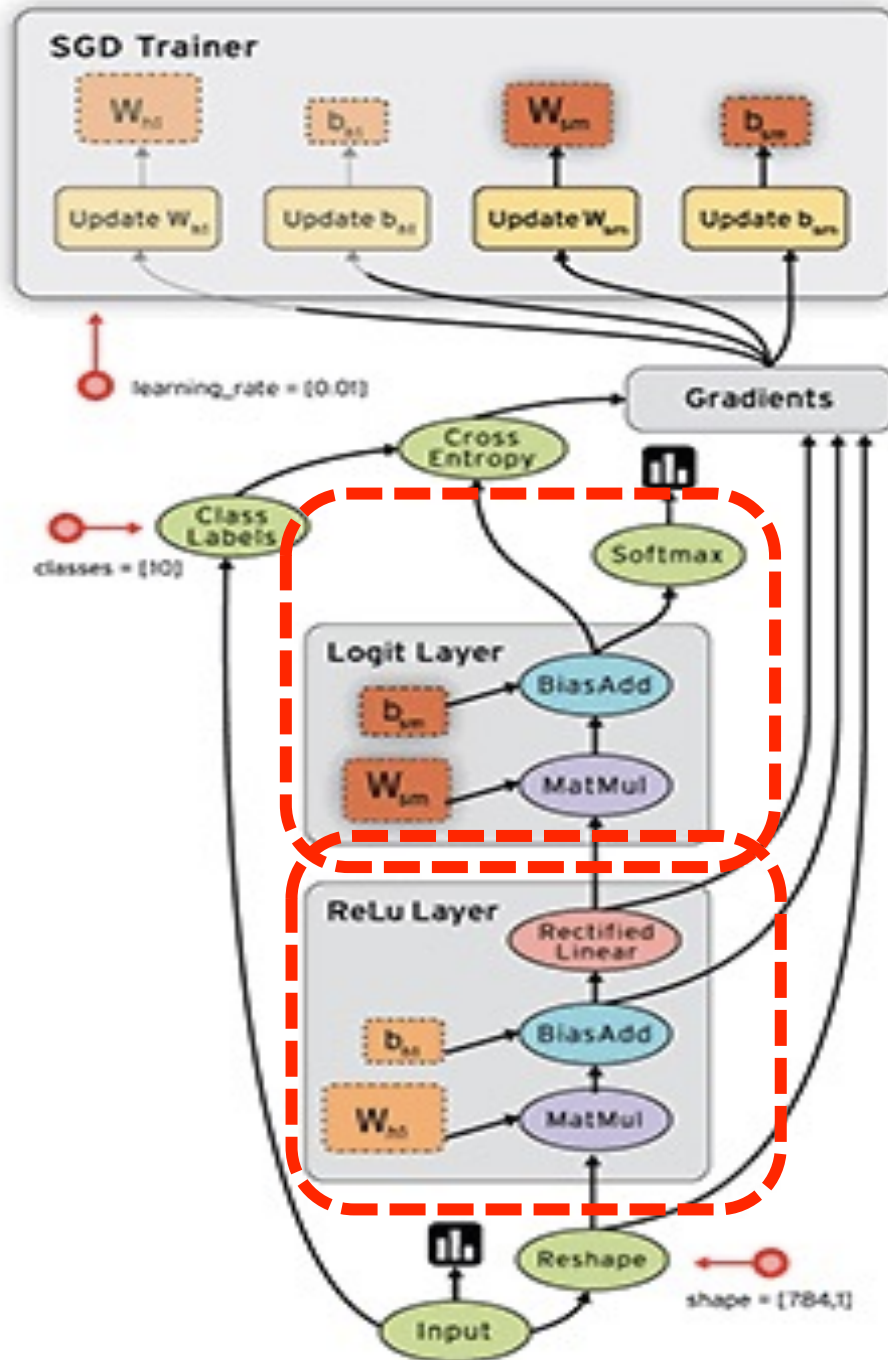
<https://www.tensorflow.org/>

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。)

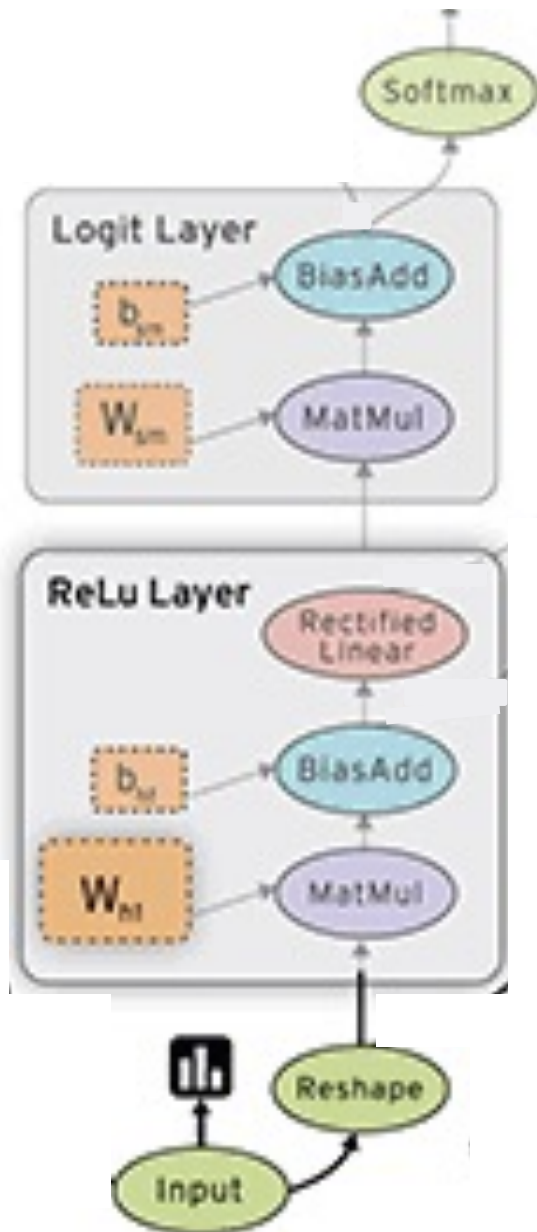


グラフの例

(学習済み)

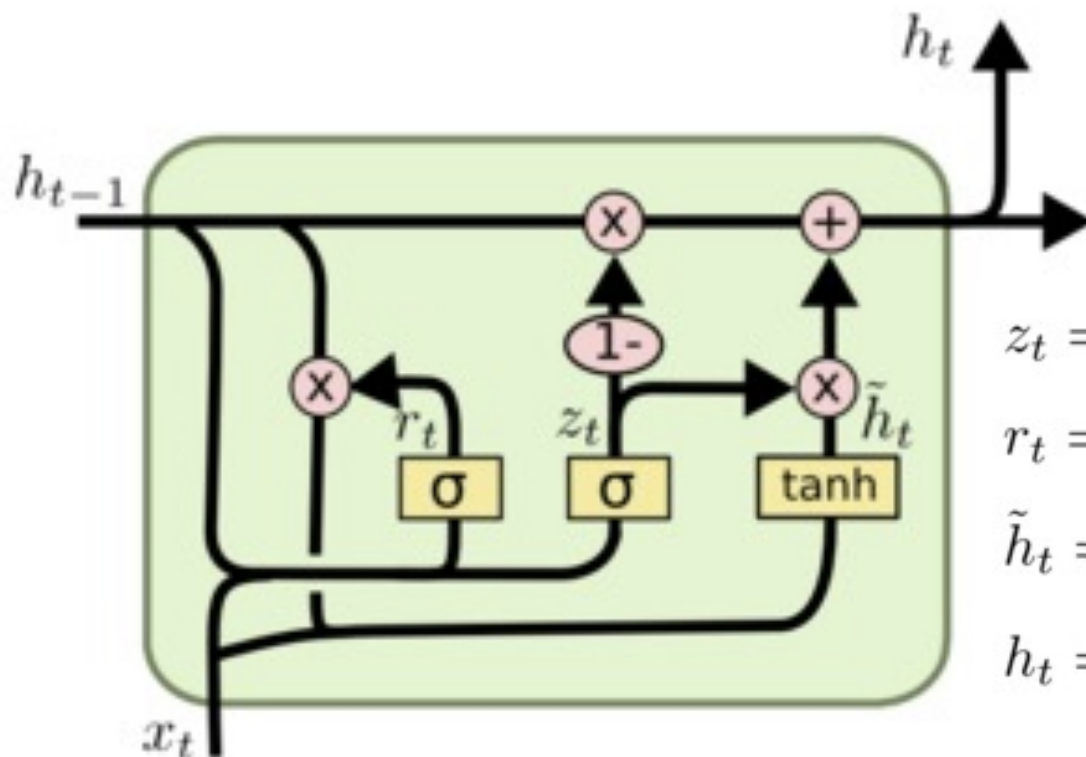
ニューラル・ネットワークでは、学習の過程と、学習済みの結果を利用する過程では、計算量には、大きな差がある。

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

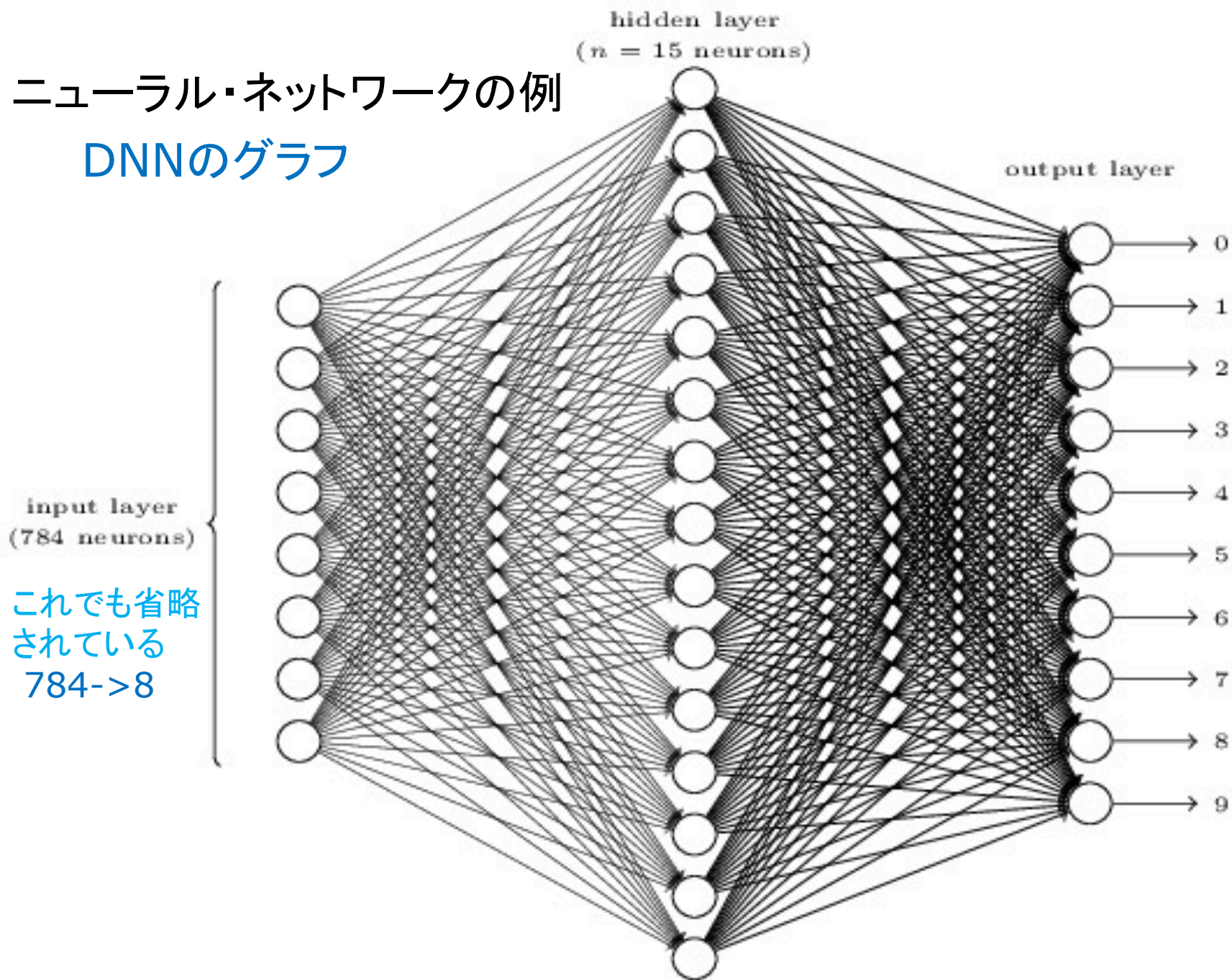
それぞれのフレームワークごとに表現のスタイルには違いがあるものの、重要なことは、それが表現しているニューラル・ネットワークの「グラフ」は、同一のものであるということである。それは、基本的なモジュールの組み合わせでできている。

フルコネクトDNNの グラフ表現とモジュール構造

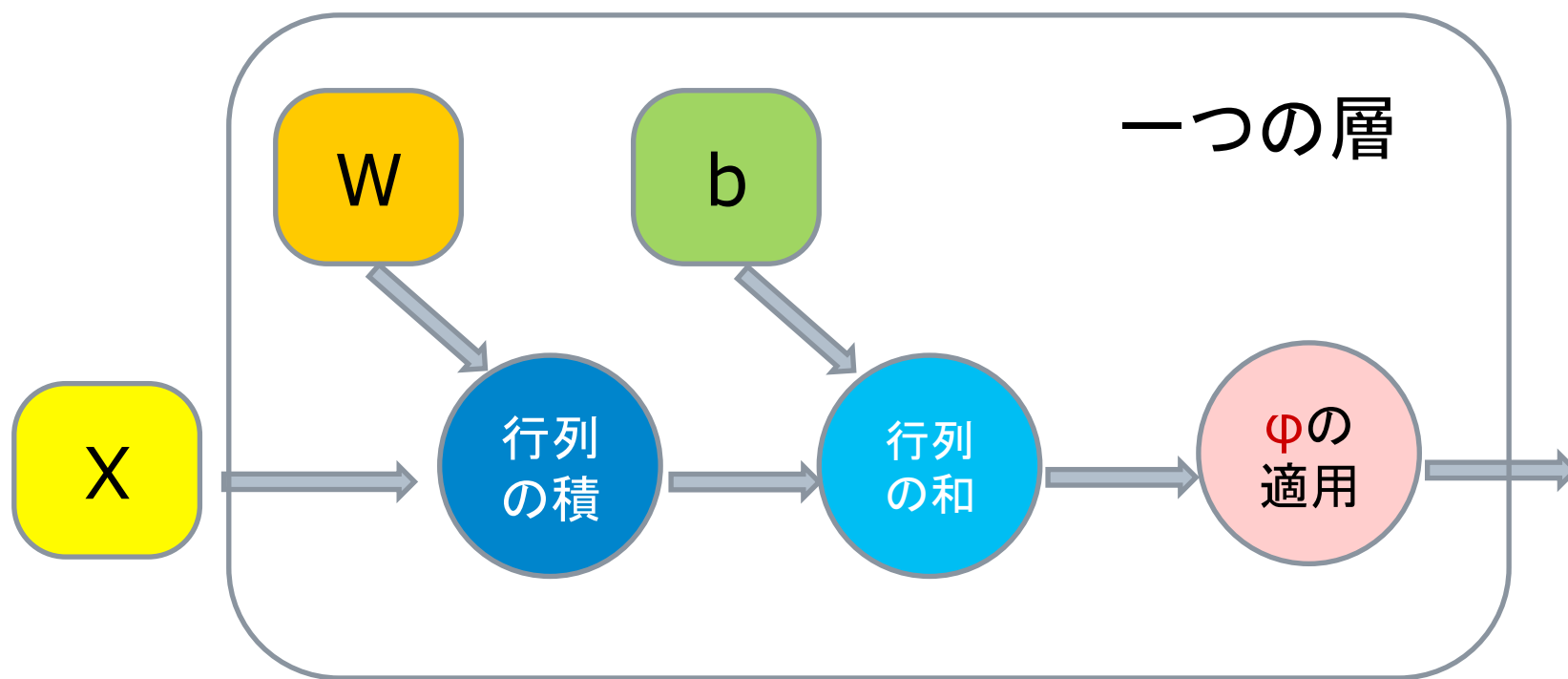
まずは、基本的な フルコネクトDNN (Deep Neural Network)のグラフ表現とモジュール構造を見ていこう

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。



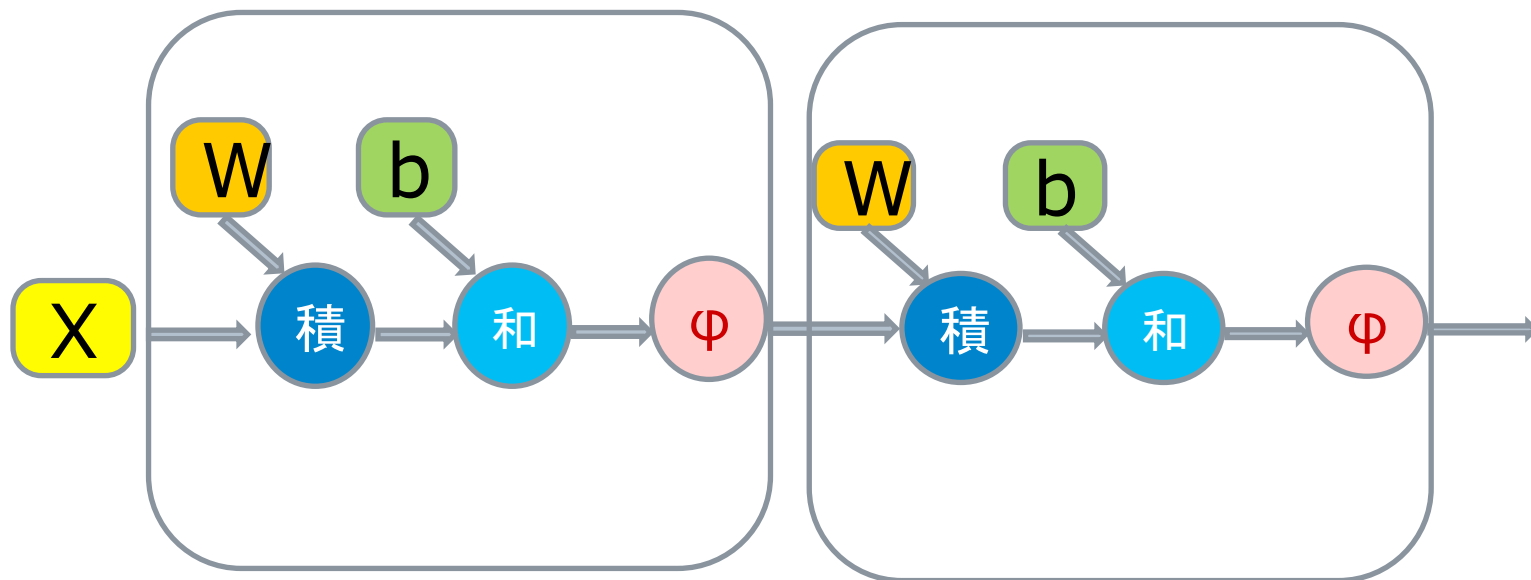
5つのノードがあるが、Wとbには、重みとバイアスの値が入り、残りの3つのノードは、演算を行うノードである。この層の働きは、 $\phi(X \cdot W + b)$ で表現される。

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



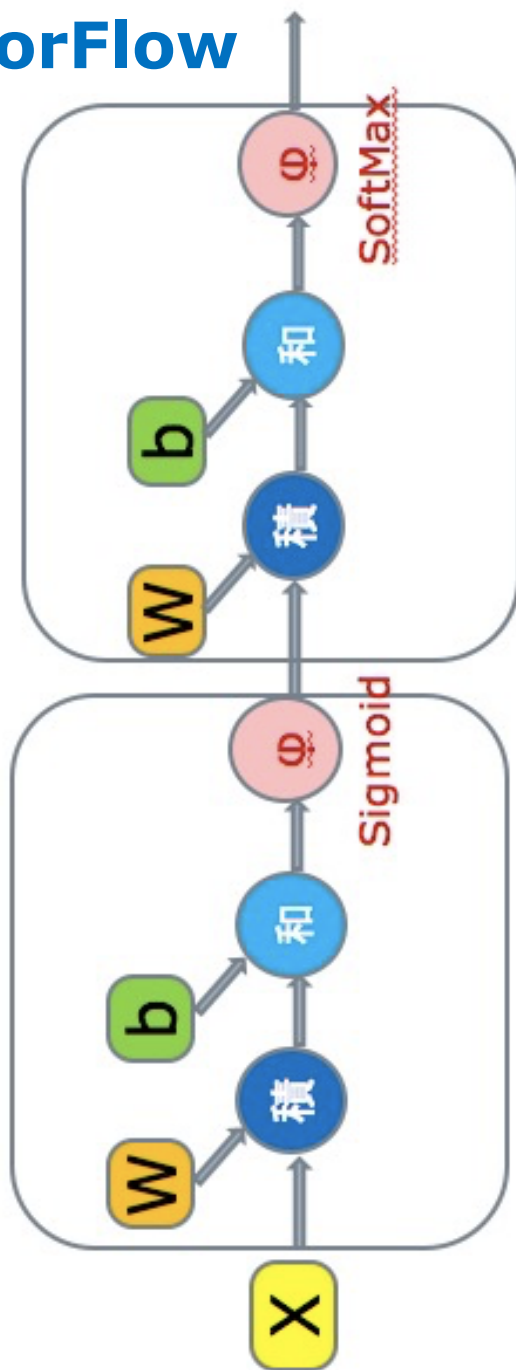
入力層

隠れ層

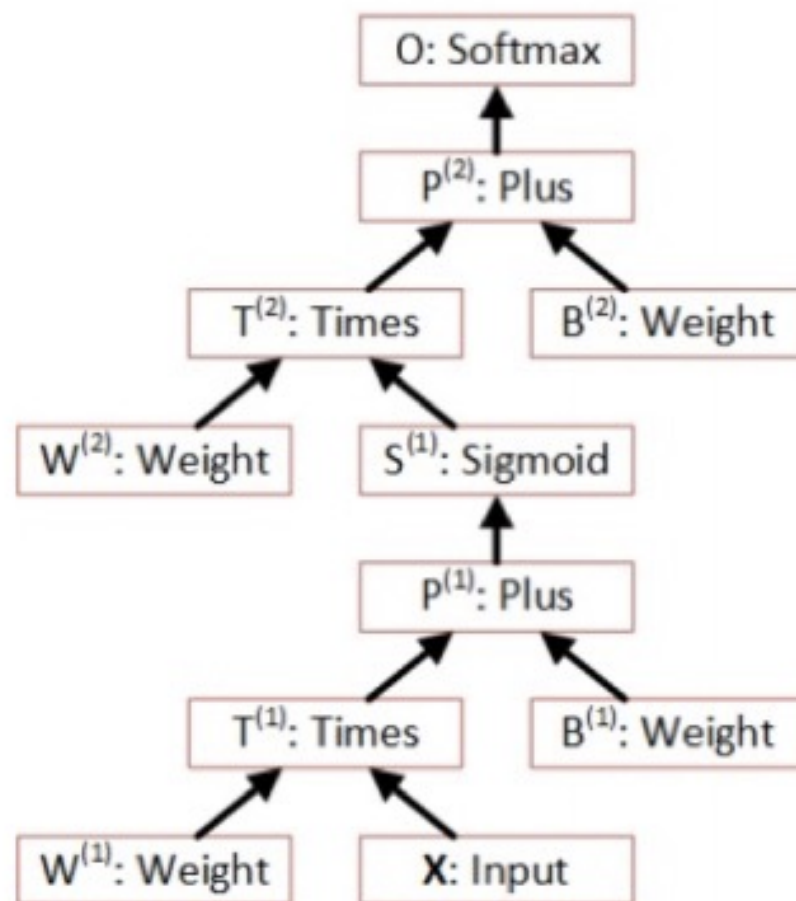
出力層

こっちのグラフの方が、ずっとわかりやすい！

TensorFlow

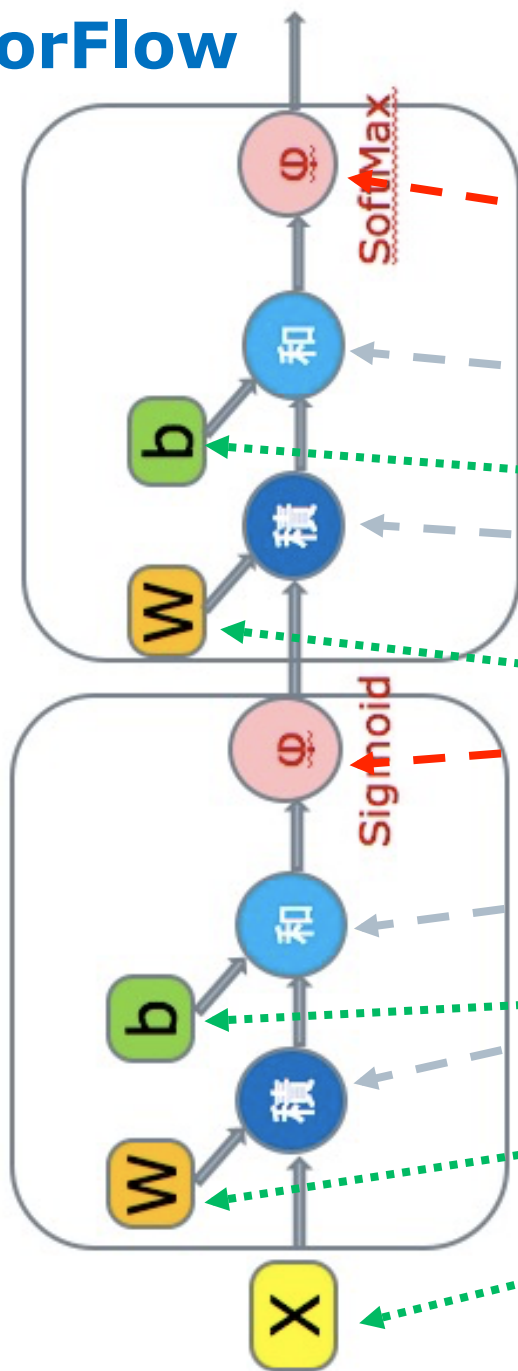


CNTK

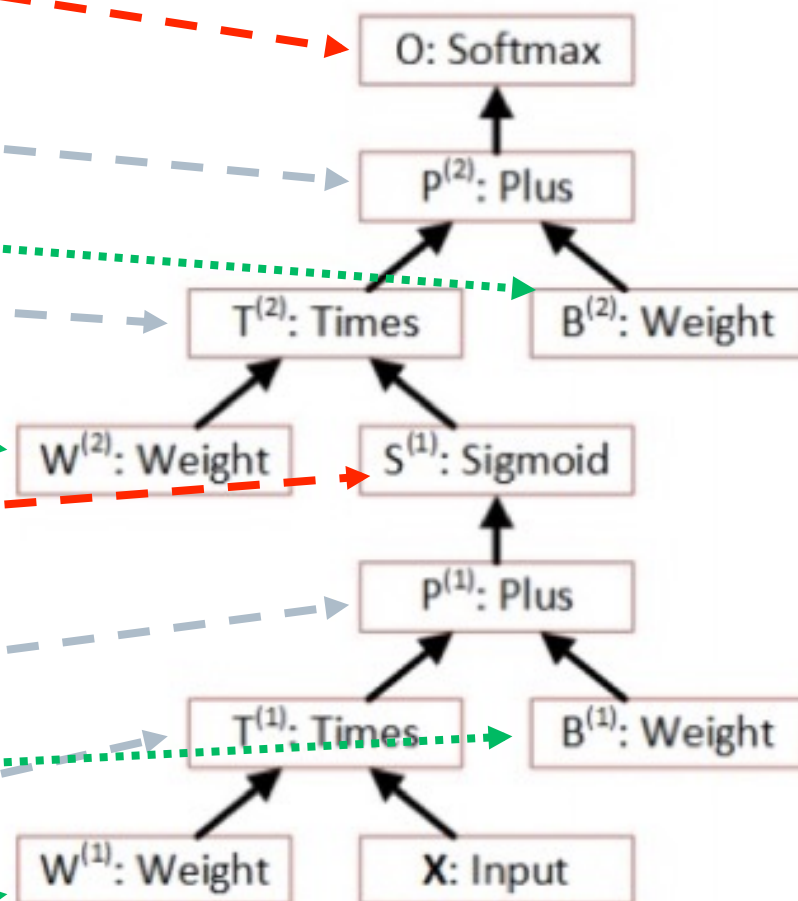


これらのグラフは、同じものである

TensorFlow

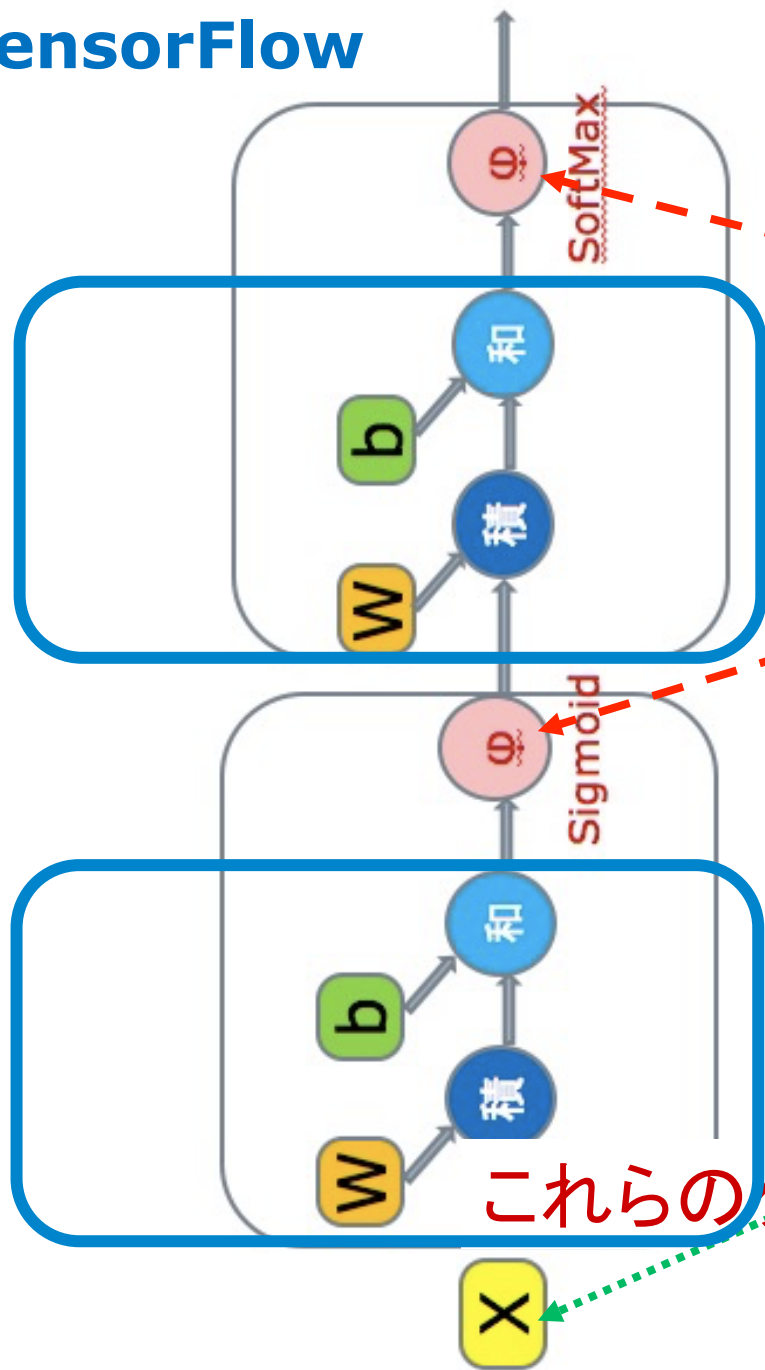


CNTK

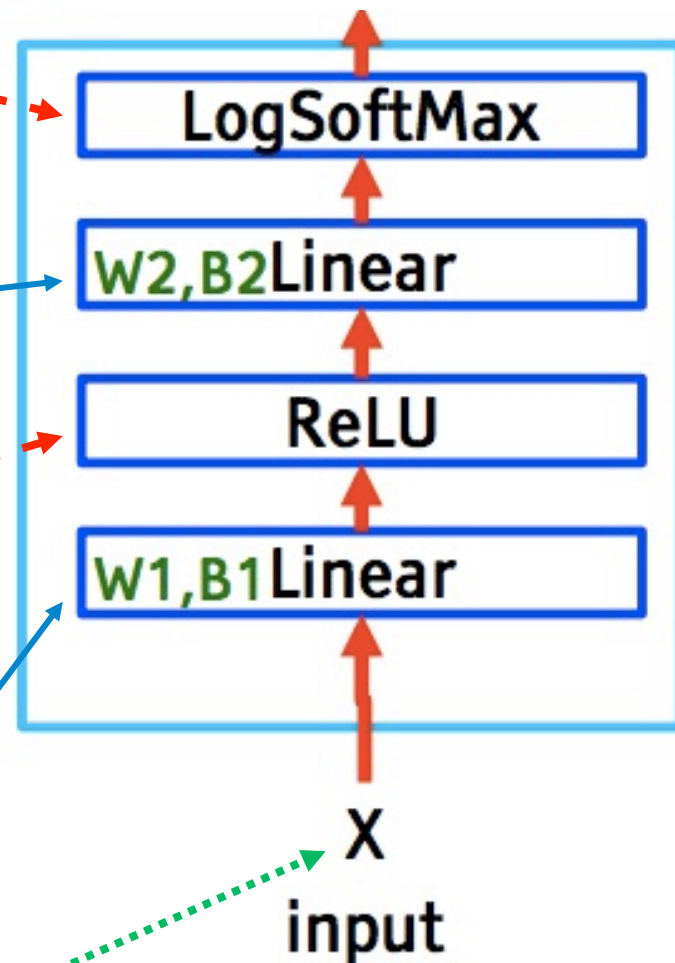


これらのグラフは、同じものである

TensorFlow



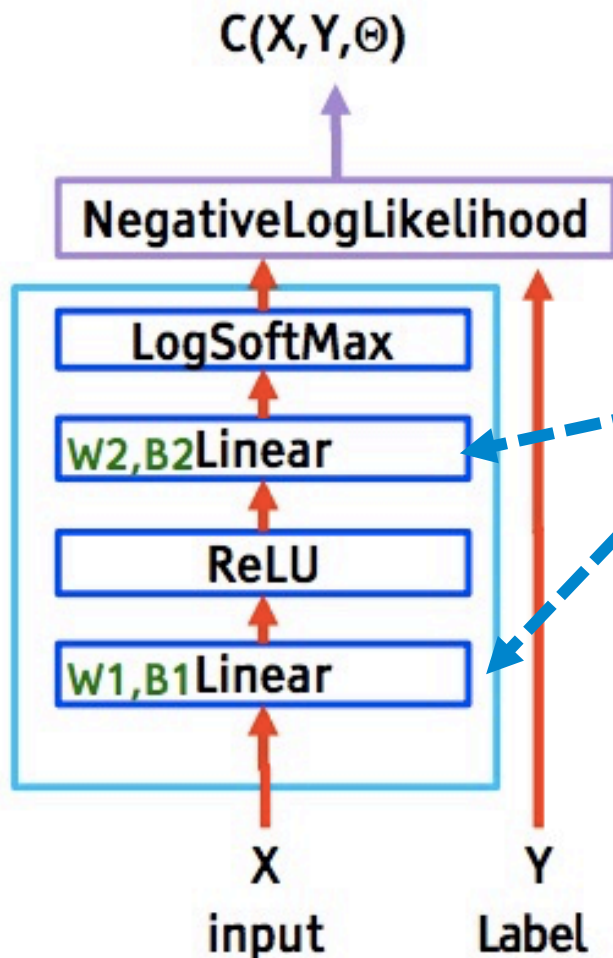
Torch7



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。

CNTKでのDNNモジュールの記述例

```
DNN = [  
    hiddenDim = 200
```

```
# DNN SigmoidLayer and DNN Layer are defined in Macros.ndl  
h1 = DNN SigmoidLayer(featDim, hiddenDim, featScaled, 1)  
ol = DNN Layer(hiddenDim, labelDim, h1, 1)
```

```
ce = CrossEntropyWithSoftmax(labels, ol)  
err = ErrorPrediction(labels, ol)
```

```
# Special Nodes  
FeatureNodes = (features)  
LabelNodes = (labels)  
CriterionNodes = (ce)  
EvalNodes = (err)  
OutputNodes = (ol)
```

```
]
```



DNN SigmoidLayer, DNN Layerの定義は、Macro.ndlの中にある。

CNTKマクロでのDNNモジュールの記述例

```
DNNLayer(inDim, outDim, x, parmScale) = [  
    W = LearnableParameter(outDim, inDim,  
        init="uniform", initValueScale=parmScale)  
    b = LearnableParameter(outDim, 1,  
        init="uniform",initValueScale=parmScale)  
    t = Times(W, x)  
    z = Plus(t, b)  
]
```

```
DNNSigmoidLayer(inDim, outDim, x, parmScale) = [  
    W = LearnableParameter(outDim, inDim,  
        init="uniform", initValueScale=parmScale)  
    b = LearnableParameter(outDim, 1,  
        init="uniform",initValueScale=parmScale)  
    t = Times(W, x)  
    z = Plus(t, b)  
    y = Sigmoid(z)  
]
```

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。



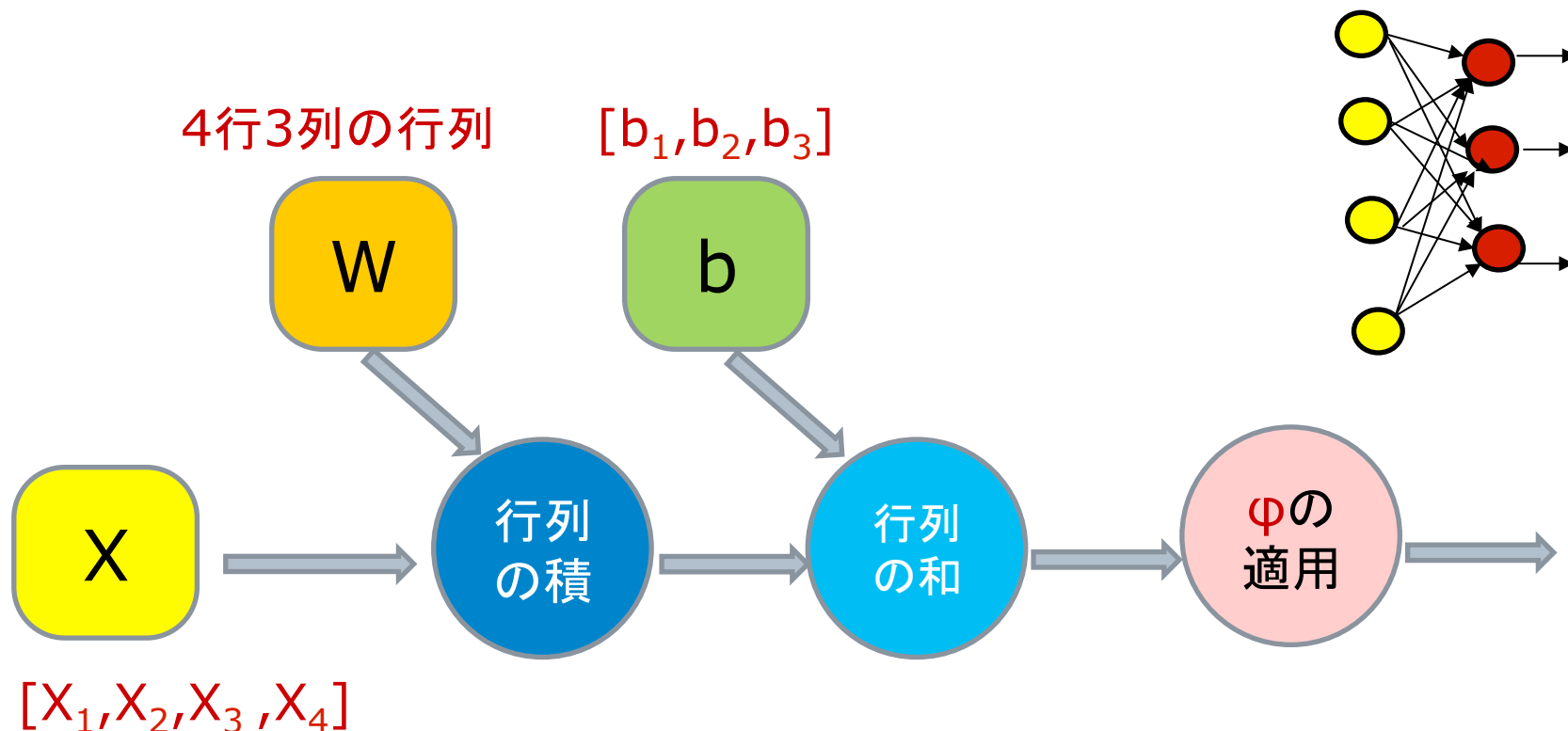
グラフを流れる量 -- テンソル

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？

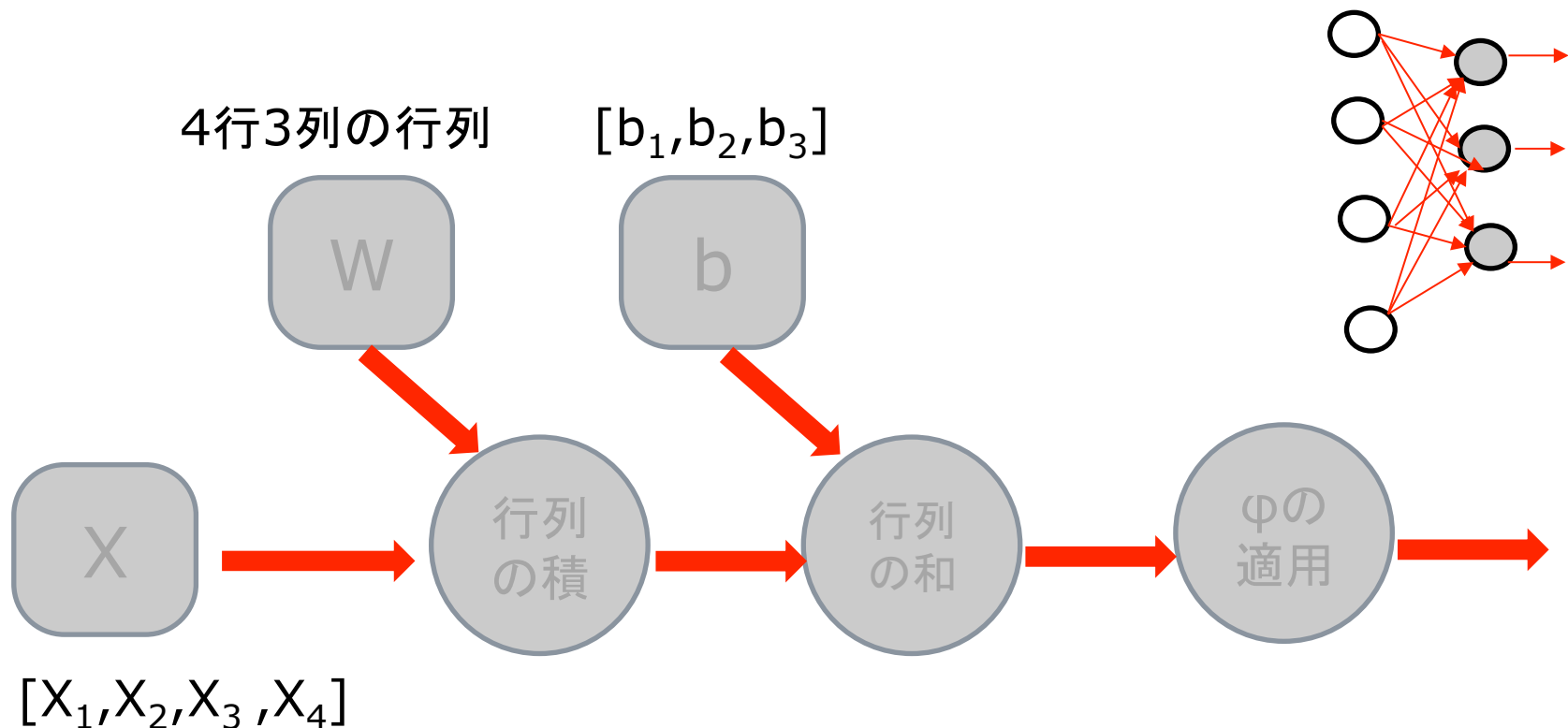
TensorFlowのグラフでは、グラフを流れるデータは、すべて「テンソル」にまとめられている。

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。



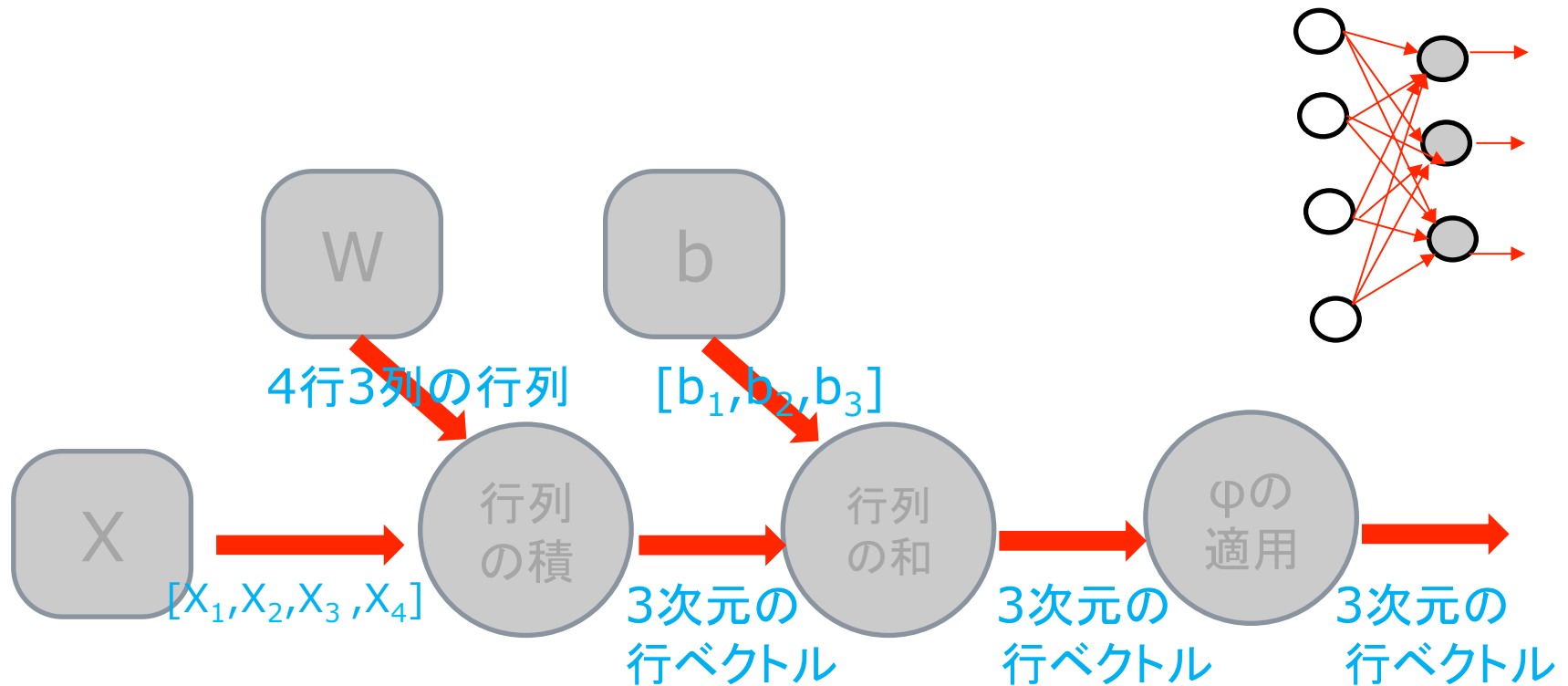
ただ、ここでは、ノードではなく辺の方に注目してみよう。
どのようなタイプのデータが、辺の上を流れるかを考えよう。

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」



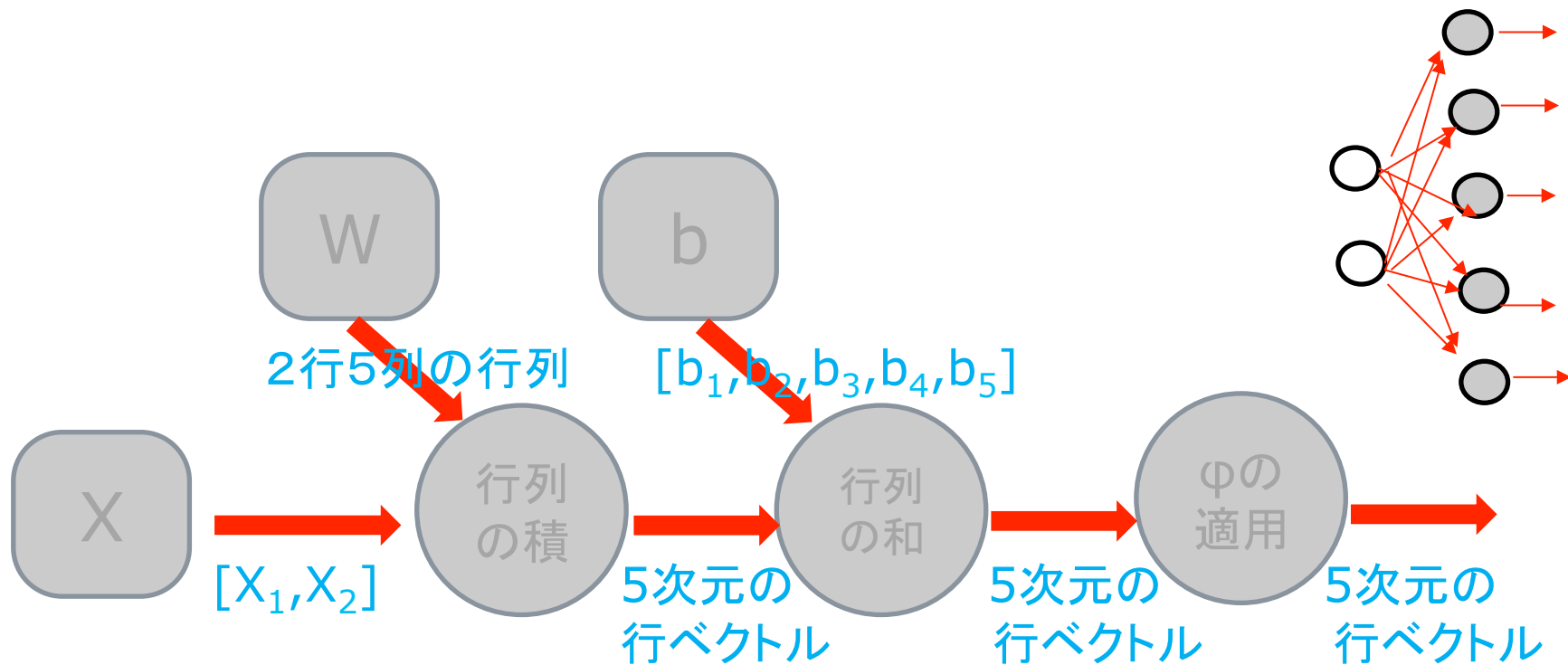
TensorFlowでは、こうしたデータをすべて**テンソル**と呼んでいる。
どのような、タイプのデータか見てみよう。

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].