

プログラミング勉強会

やってみようぜ、ディープラーニングのプログラミング

~~7月22日改訂版~~ **9月17日 更に改訂版**

~~2017年5月28日~~

名古屋校 2011期 越智 由浩

自己紹介



越智 由浩（おち よしひろ）

愛媛生まれの43歳。祖父母・両親・姉 ほぼ皆 美容師
鎌倉/東京育ち、名古屋在住、5人家族



1998年～日本アイ・ビー・エム

2012年～ネットアップ 名古屋支店長

2016年～ 〃 トヨタ担当 グローバル・マネージャ



2010年7月から半年間は単科、

2011年4月から大学院へ

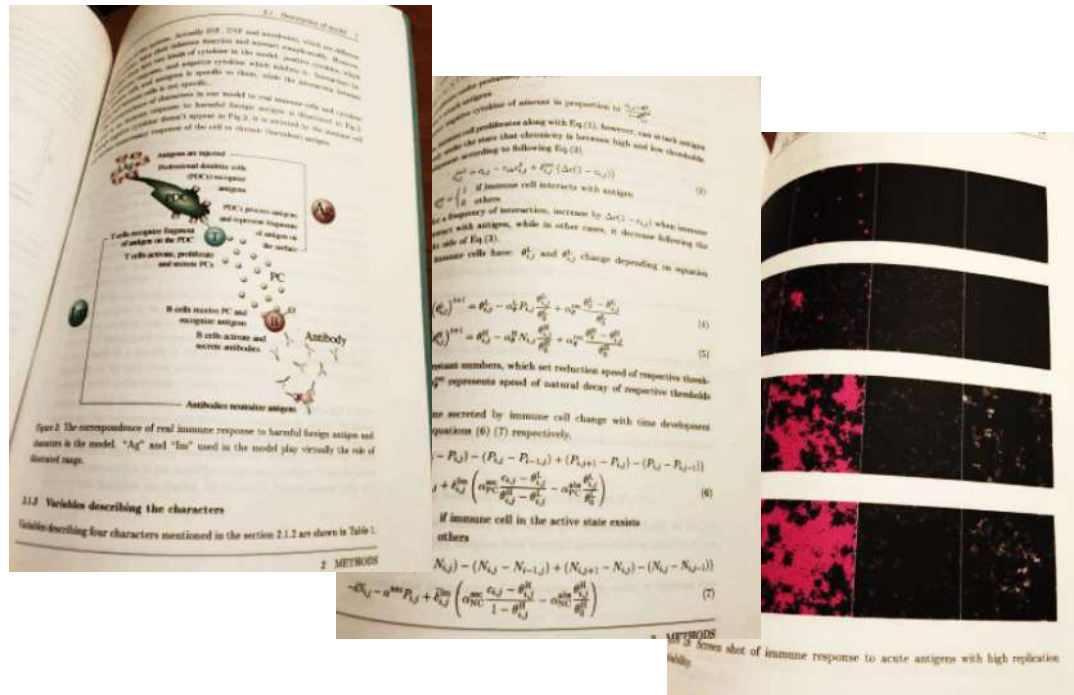
2014年4月、卒業

私とプログラミング

小学生の頃～



学生の時
カオス・複雑系
免疫系のコンピューター・シミュレーション



最近
オタクっぽい趣味の再燃



今日にいたる経緯 ～名古屋校 プログラミング勉強会 クラブ活動～

メモ：リンクあり



**9月
25日** プログラミング ハンズオン&ワークショ...
日 14:30 - グロービス 名古屋校 101教室
プログラミング勉強会主催



**10月
30日** 実践！Excel VBA入門/プログラミング ...
日 14:30 - グロービス 名古屋校 101教室
プログラミング勉強会主催



**11月
27日** *IoTコトハジメ* 世界でたった一つだけ...
日 14:00 - グロービス 名古屋校 101教室
プログラミング勉強会主催



**2月
19日** 野呂さん公認・パクリ版*ゼロからのR...
日 14:00 - グロービス名古屋 101教室
プログラミング勉強会主催



**4月
8日** ここ中部圏をプログラミング教育のメッ...
土 14:00 - グロービス名古屋校 101教室
プログラミング勉強会主催



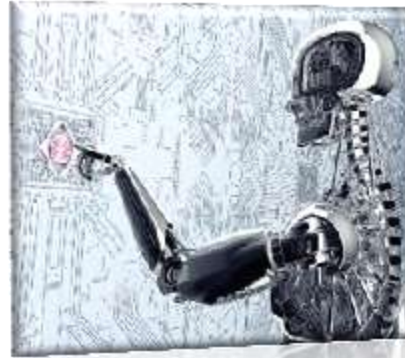
**5月
28日** やってみようぜ、ディープラーニングの...
日 14:00 - グロービス名古屋校 101教室
プログラミング勉強会主催

ここから本題

今日の立ち位置 ～ 人工知能を学ぶ中で

社会・ビジネス視点

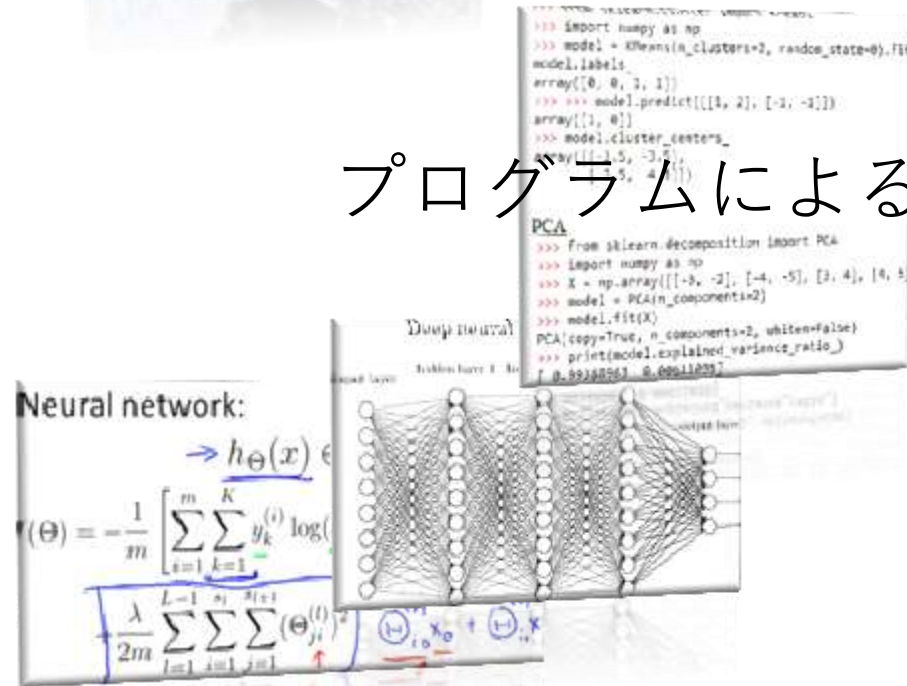
ディープラーニングで何ができるのか、世の中がどう変わるのか



コトバノチガイ

サイエンス・テクノロジー視点

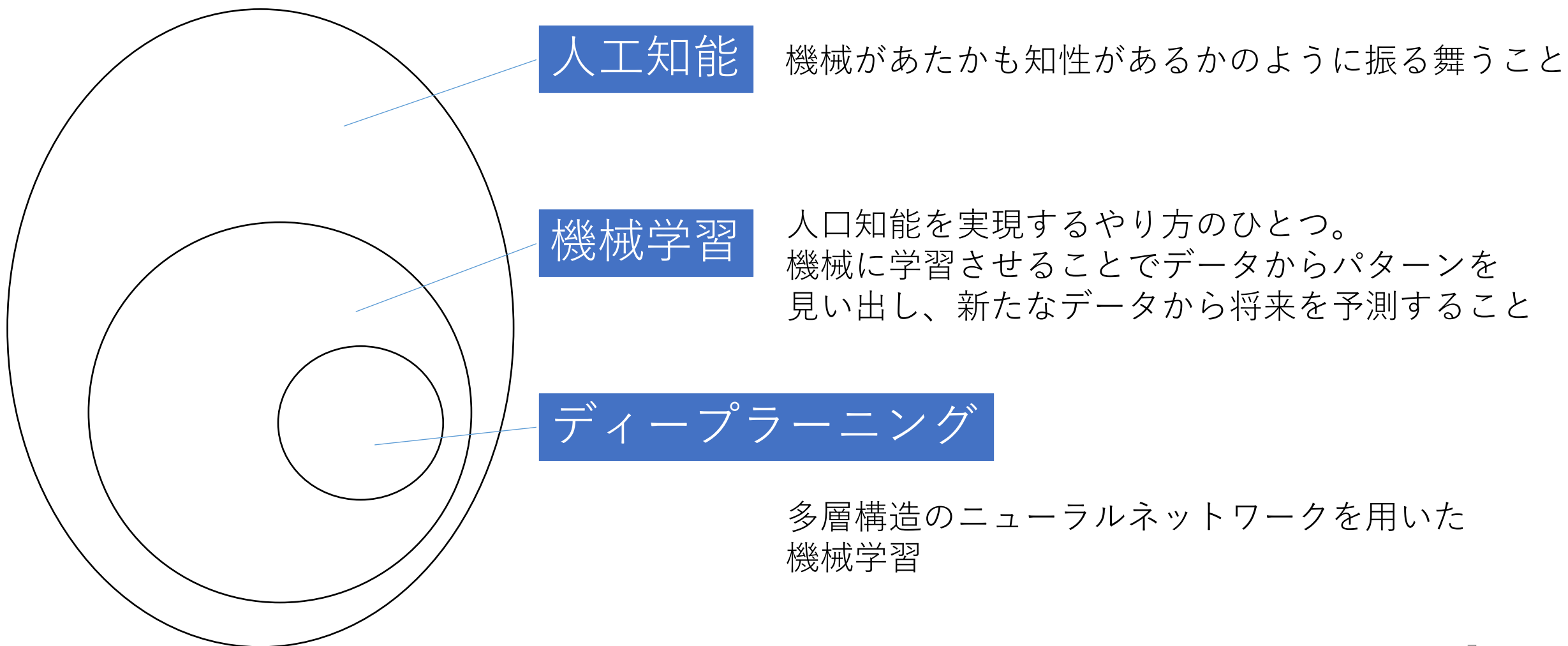
ディープラーニングってそもそも中身は何をやっているのか



プログラムによる実装

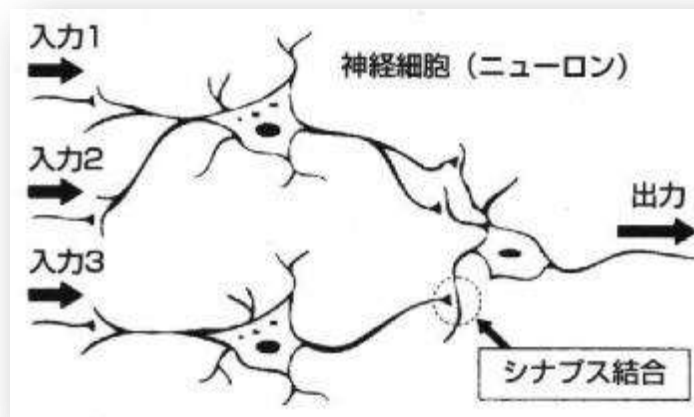
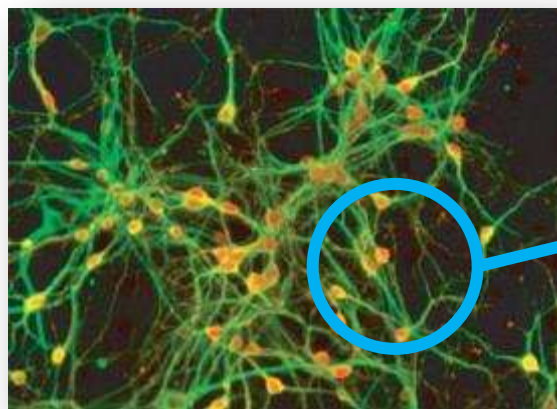
今日はこちら側の基礎的なところ

まず、言葉の整理



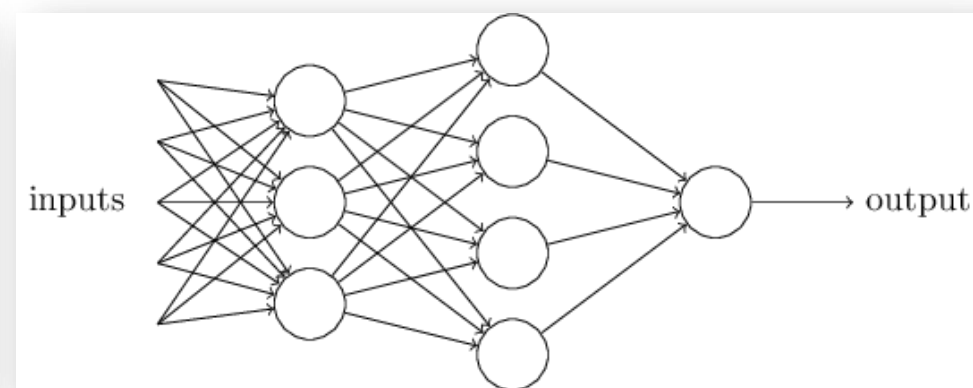
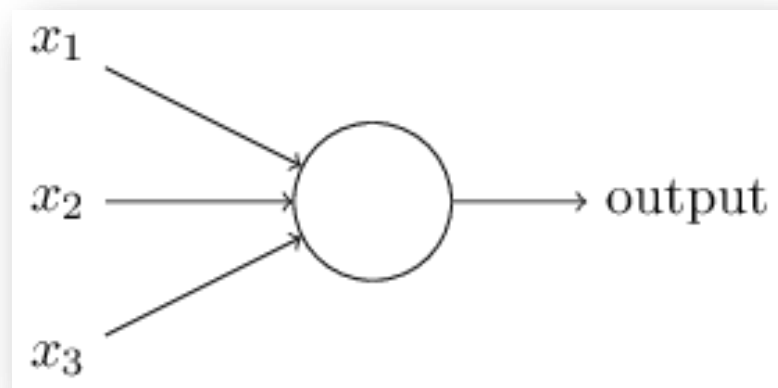
ニューラルネットワークモデル

実物



抽象的に表現

計算できる"モデル"

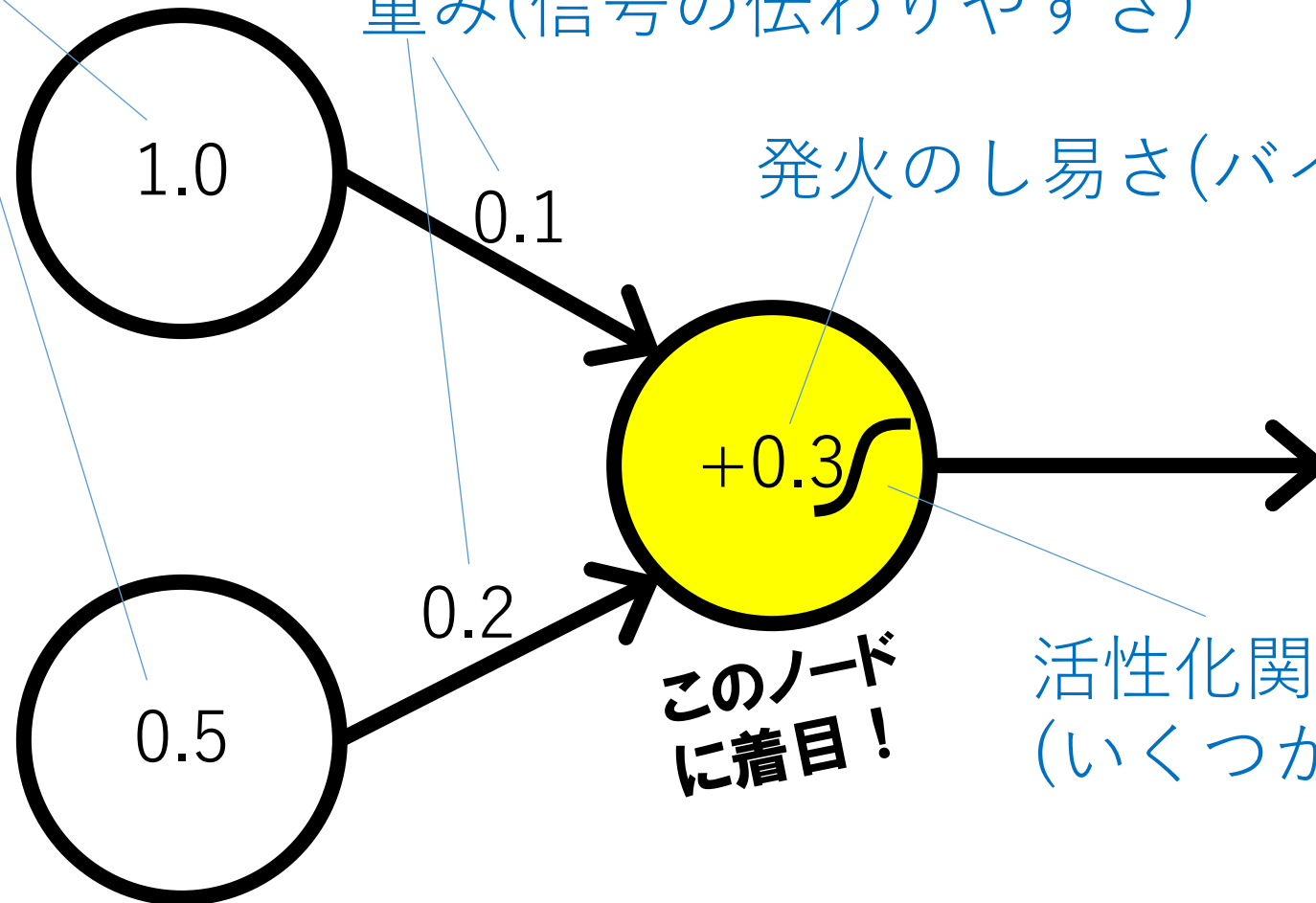


ニューラルネットワークモデルの計算 ルール

入力元の信号の強さ

重み(信号の伝わりやすさ)

発火のし易さ(バイアス)

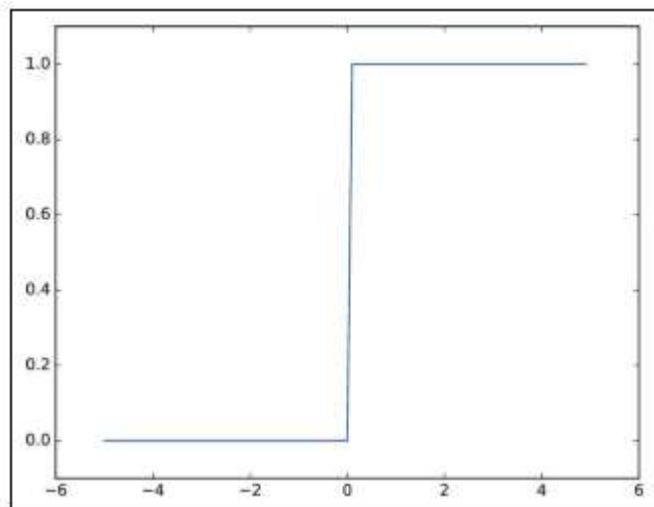


活性化関数
(いくつか種類がある)

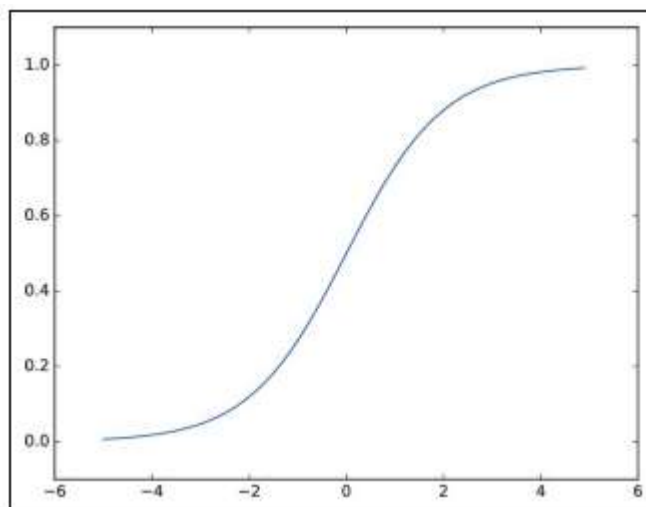
メモ：関数について補足

活性化関数

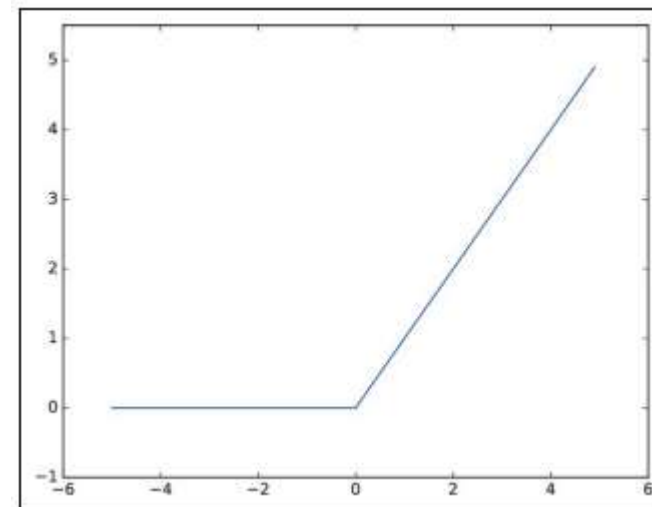
ステップ関数



シグモイド関数



ReLU関数
(Rectified Linear Unit)



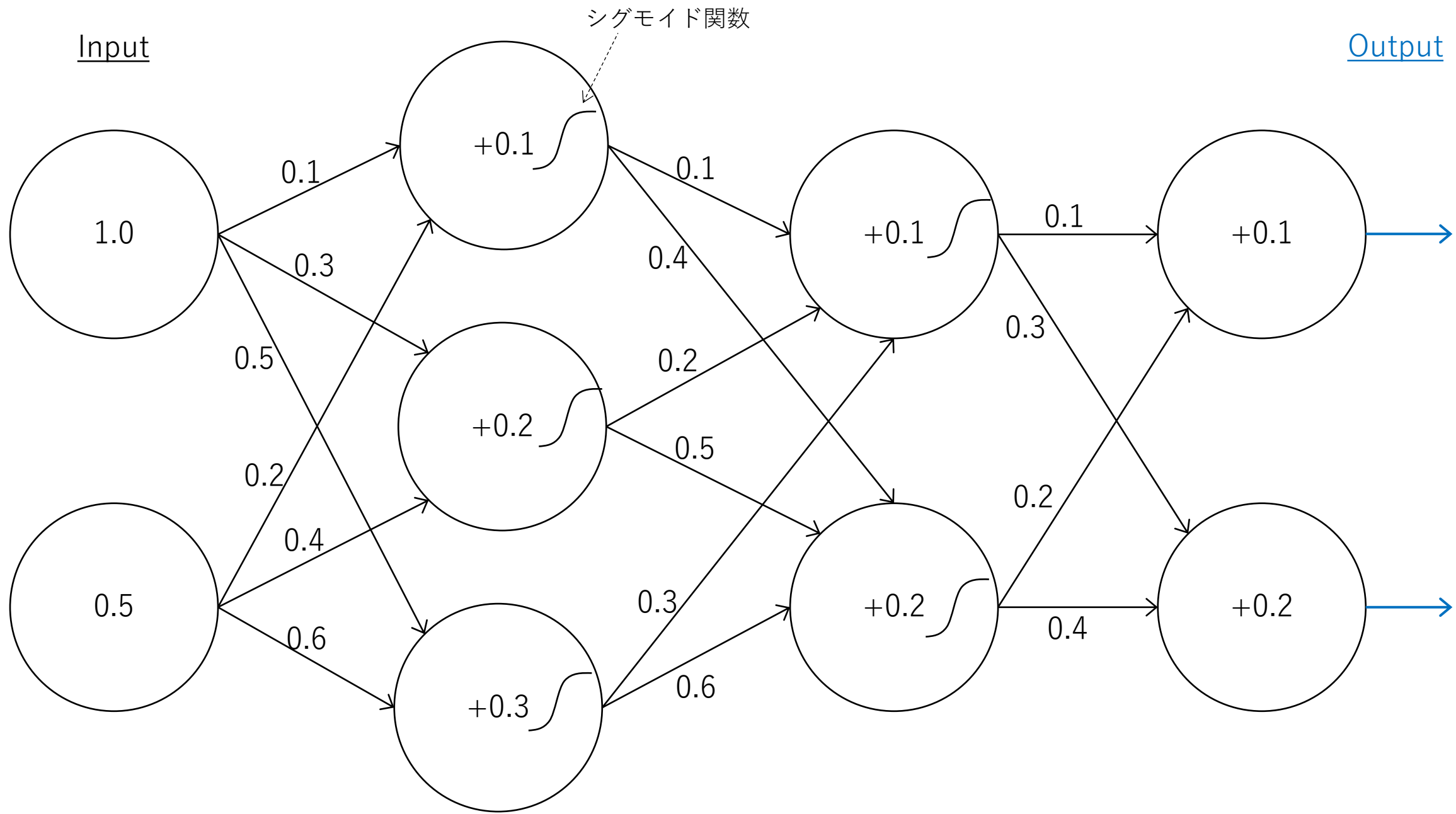
出力
↑
入力の総和 + バイアス
→

$$h(x) = \begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases}$$

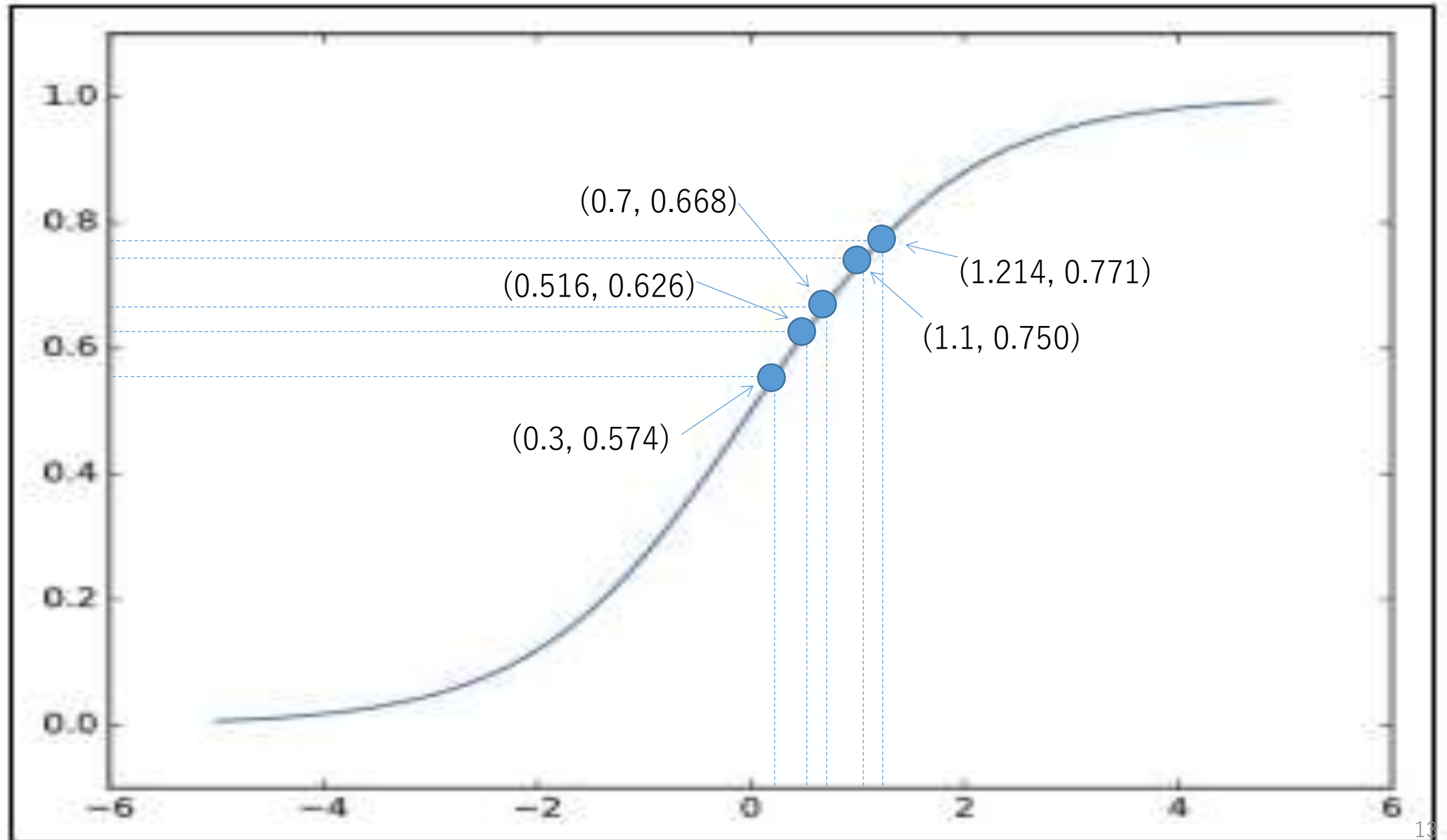
$$h(x) = \frac{1}{1 + \exp(-x)}$$

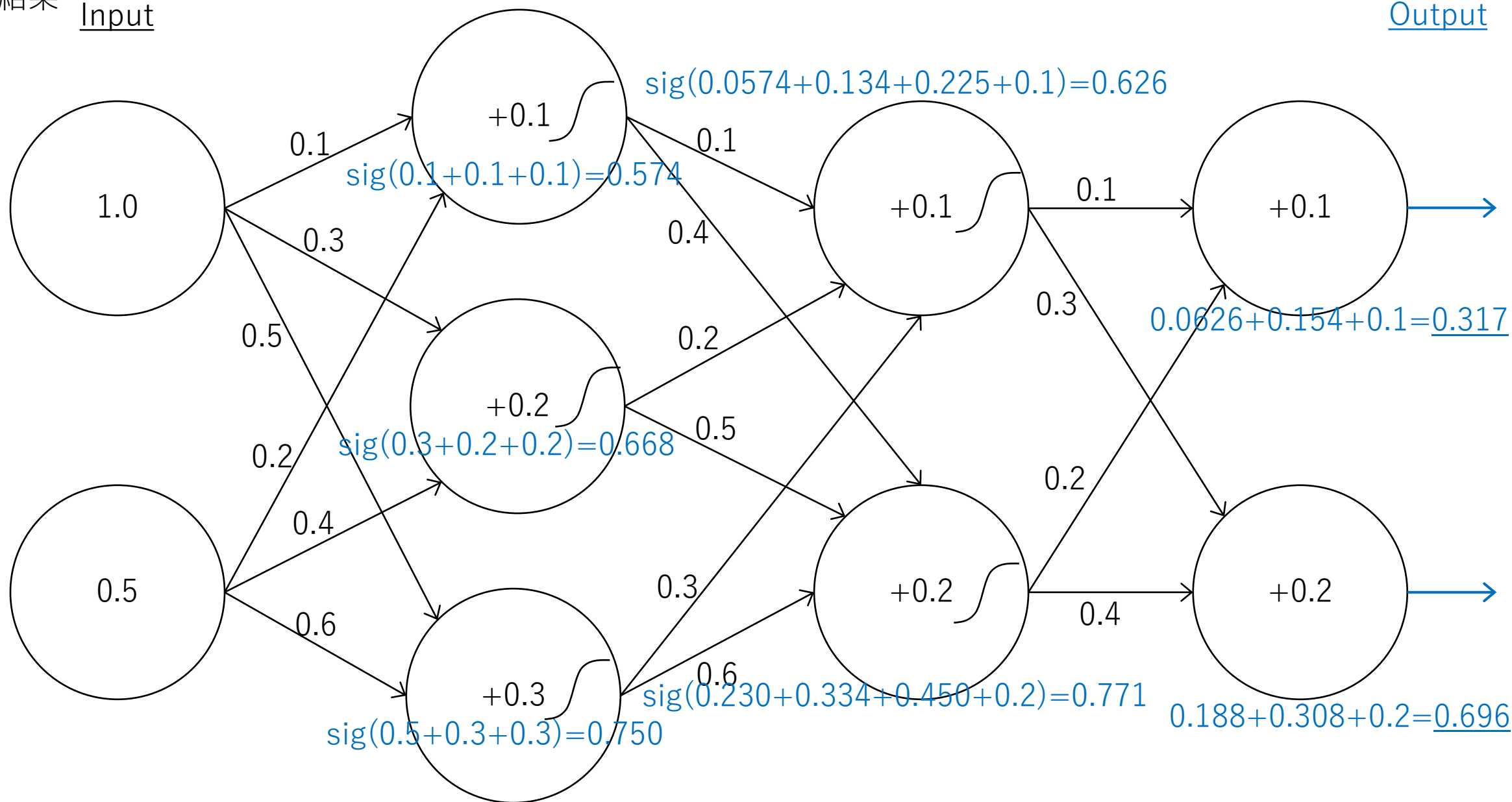
$$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$

手触り感持って”模型”と慣れ親しむために
手で計算してみる



シグモイド関数、参照シート



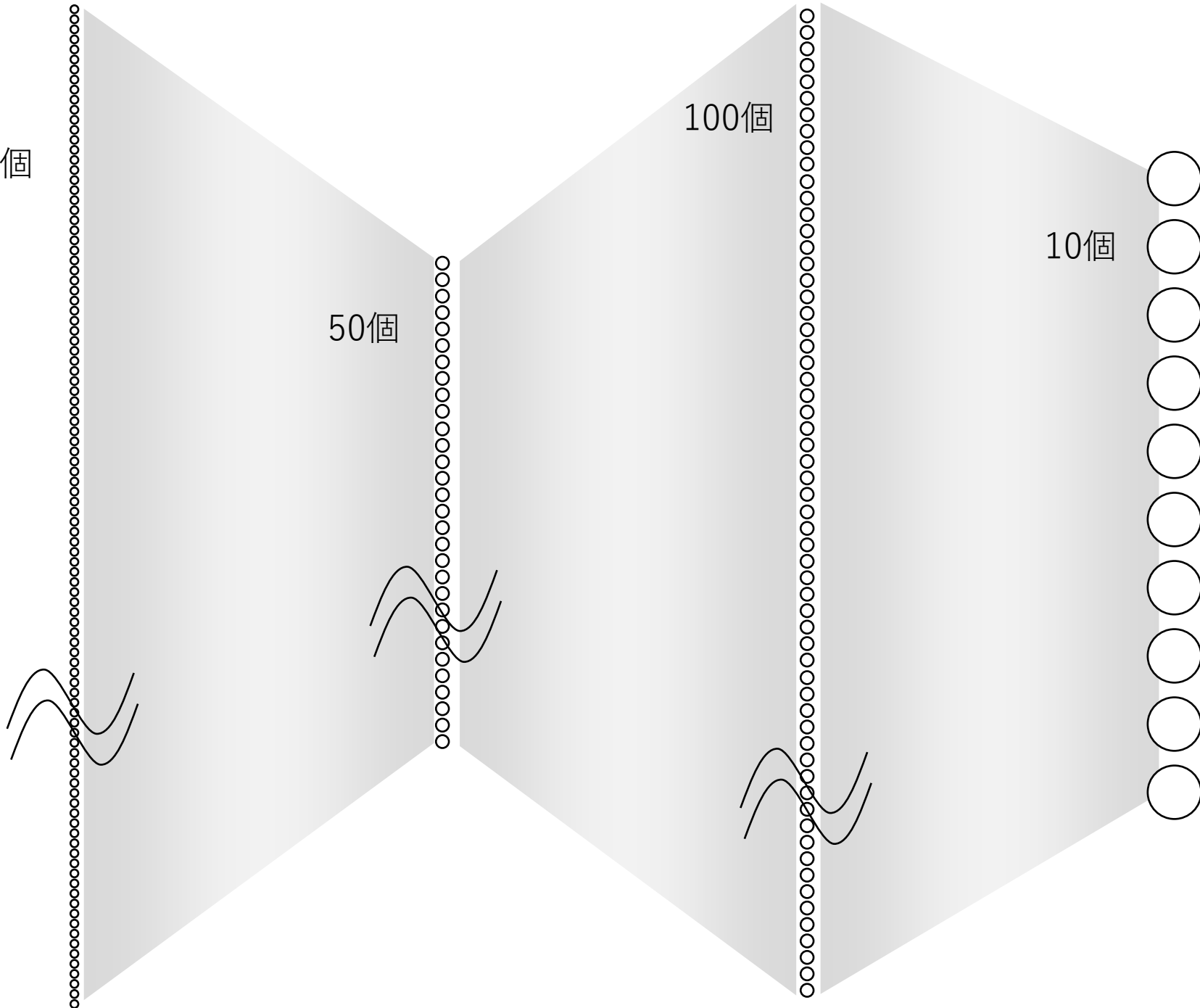
InputOutput

784個

50個

100個

10個



なぜ、こんなことをするのか？

784個

入力

50個

計算

(膨大な量の
掛け算と足し算
と活性化関数)

100個

10個

出力

この”模型”によって0~9の数字の手書き文字認識ができてしまう！

手計算じゃムリ！



計算
(膨大な量の
掛け算と足し算
と活性化関数)

単調な計算の繰り返しは
コンピューターの得意技。
プログラムを作って計算
させちゃえばいい！



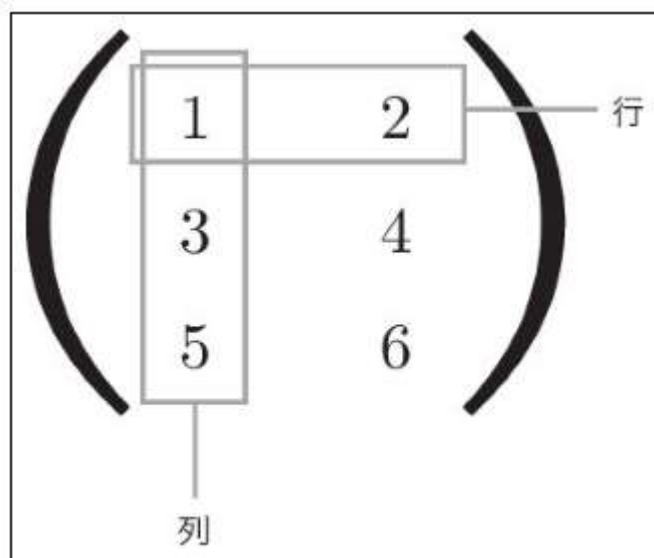
ふたつの武器を手に入れよう

- 数学の道具 – 「行列」と「行列の内積」
- プログラミング – Python（なぜPythonかは後ほど補足）

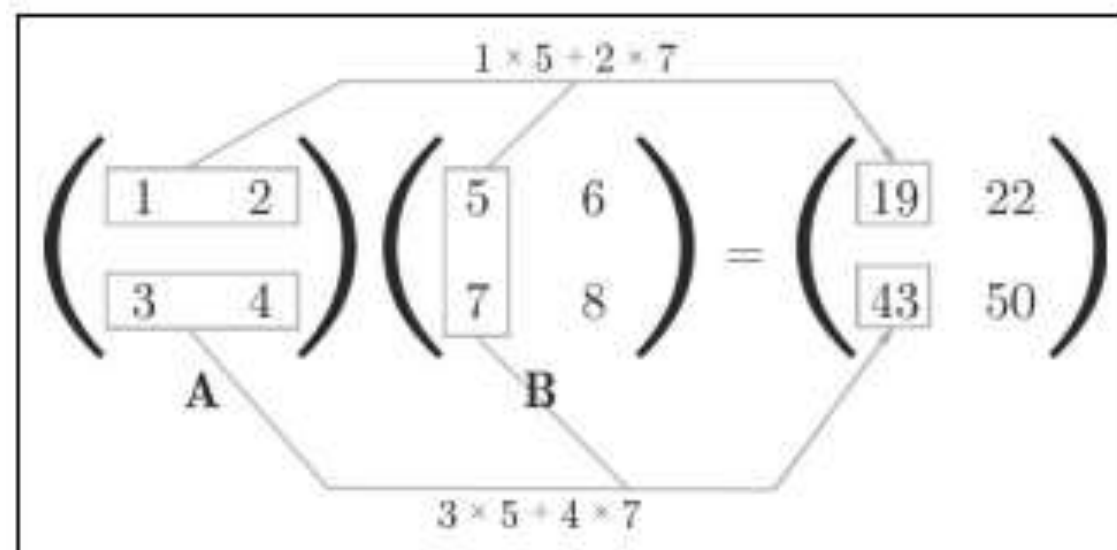
数学の便利な道具 – 行列と行列の内積

なぜ便利かは後ほどわかる

3 × 2の行列の例



行列の内積の例



行列計算に慣れ親しむ

$$\begin{pmatrix} 1 & 3 \\ 5 & 7 \\ 9 & 11 \end{pmatrix} \cdot \begin{pmatrix} 11 \\ 22 \end{pmatrix} = \begin{pmatrix} 77 \\ 209 \\ 341 \end{pmatrix}$$

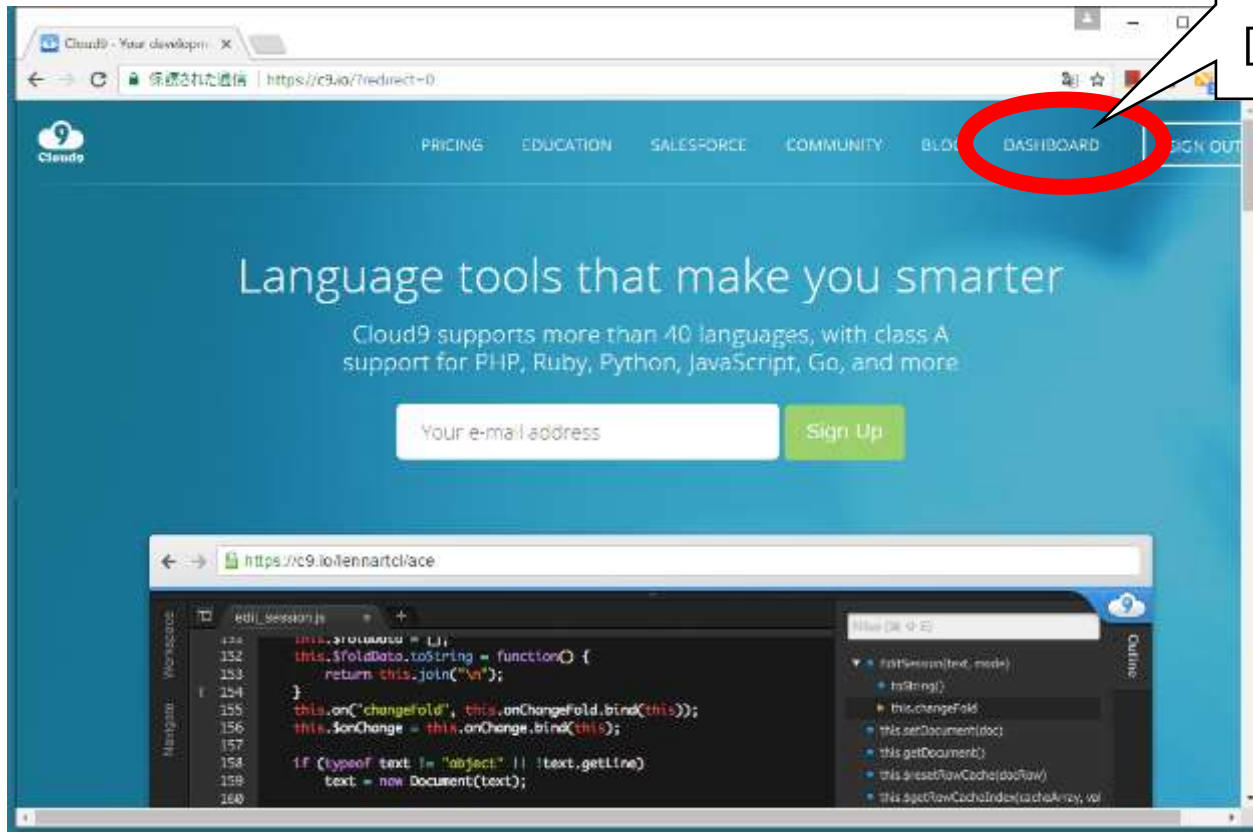
$$(11 \ 22) \cdot \begin{pmatrix} 1 & 3 \\ 5 & 7 \\ 9 & 11 \end{pmatrix} = \boxed{?}$$



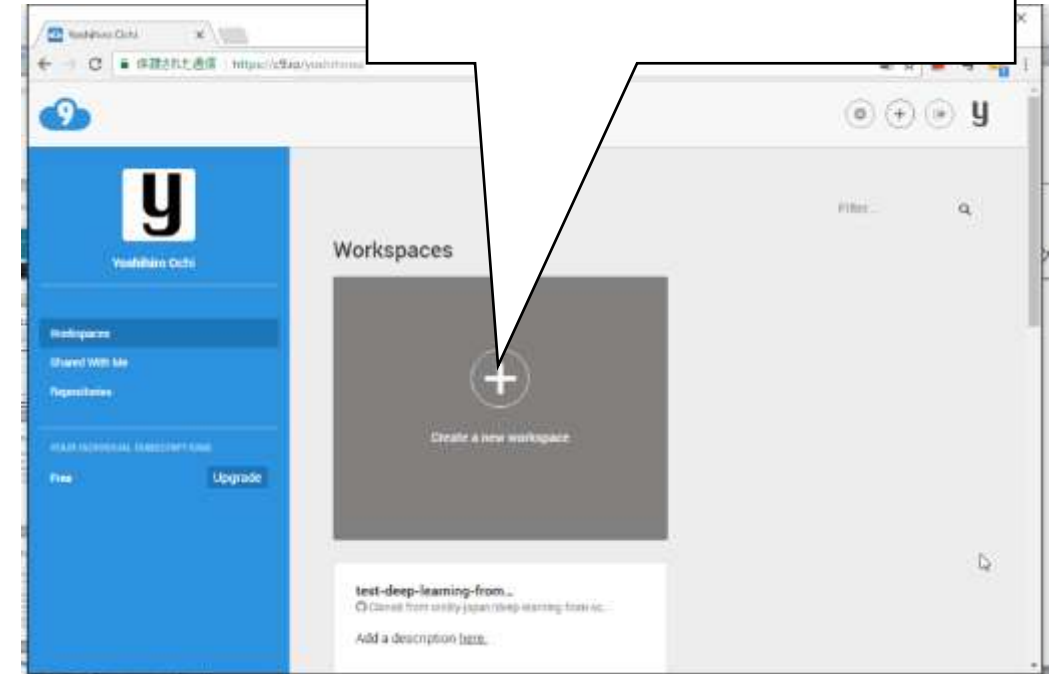
ちよつとその前に

Cloud9によるPython開発環境（１）

①Cloud9ログイン後、
トップページから
DASHBOARDをクリック



②「+」をクリックする



Cloud9によるPython開発環境（2）

The screenshot shows the 'Create a New Workspace' page on the Cloud9 website. The browser address bar shows 'https://c9.io/new'. The form has several sections: 'Workspace name' with a text input containing 'deep-learning'; 'Description' with a text input containing 'Make a short description of your workspace'; 'Hosted workspace' (selected) with tabs for 'Clone workspace', 'Remote SSH workspace', and 'Salesforce'; 'Private' (selected) with a sub-option for 'Public'; 'Clone from Git or Mercurial URL (optional)' with an empty text input; 'Choose a template' with a grid of icons for HTML5, Node.js, PHP, Python (highlighted with a blue border), Django, C++, Wordpress, Rails Tutorial, and Blank; and a green 'Create workspace' button at the bottom left.

③「Workspace name」は
適当な名前を入力

④「Clone from Git or Mercurial URL (optional)」
空白のままでOKです！

⑤「Choose a template」は
Pythonをクリックして選択

⑥Create workspaceをクリック

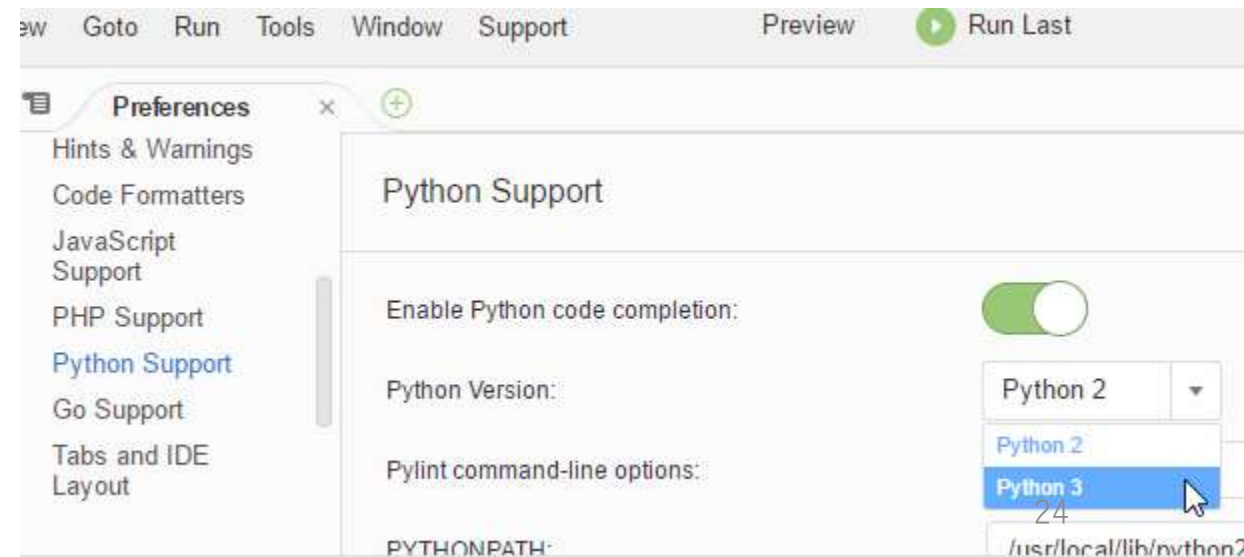
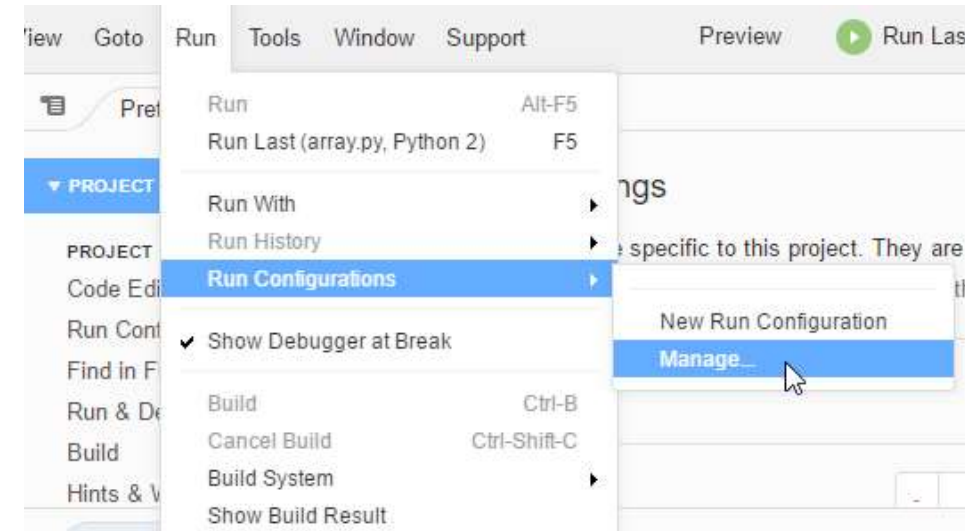
こんな画面になったら完了！→



Cloud9によるPython開発環境（3）

⑦Pythonのバージョンを3に変更

- Run -> Run Configurations -> Manage
- Preference
 - Python Support : Python Version:
 - Python 3へ変更
- Preferenceタブを閉じる



Cloud9によるPython開発環境（４）

⑧Githubから必要なファイルをダウンロード

- 下記URLに行く

<https://github.com/yoshihiroo/programming-workshop/tree/master/deep-learning>

- 0528.zip をダウンロードし、解凍すると下記 6 つのファイルができる

 Hello.py

 mnist.pkl

 mnist.py

 sample_weight.pkl

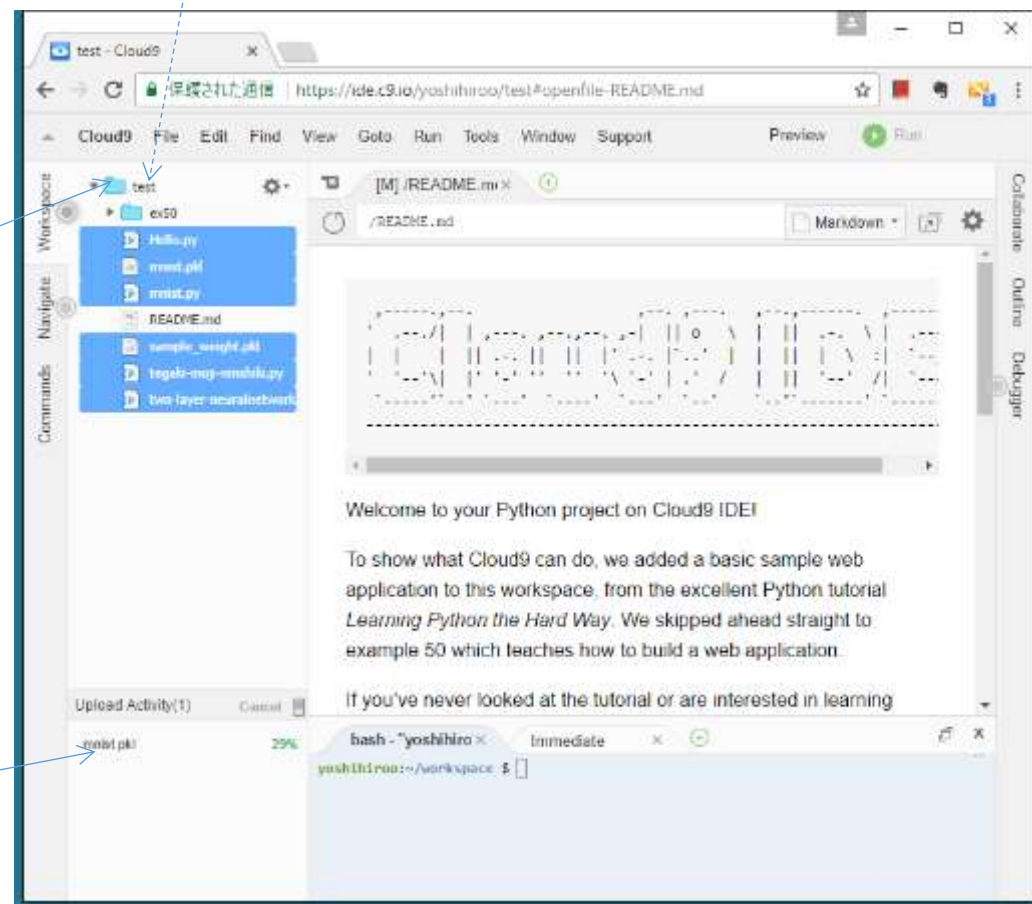
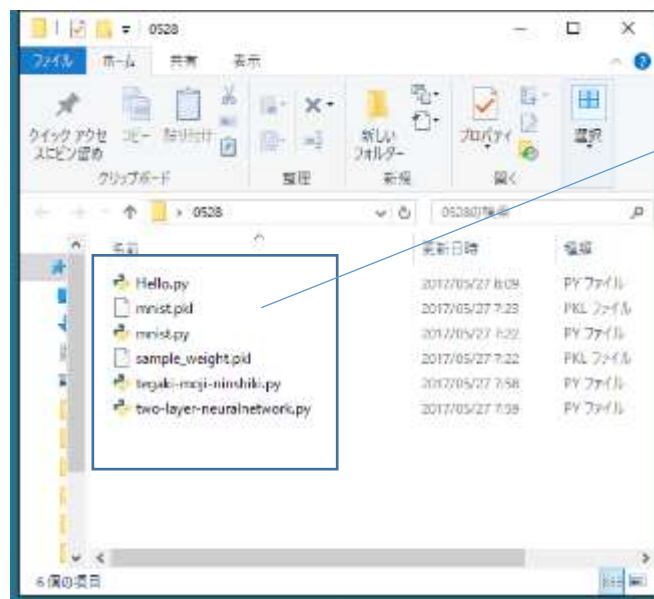
 tegaki-moji-ninshiki.py

 two-layer-neuralnetwork.py

Cloud9によるPython開発環境（5）

- ⑨Cloud9にファイルをアップロード
 - 解凍した6つのファイルをドラッグ&ドロップでCloud9上のフォルダにコピー

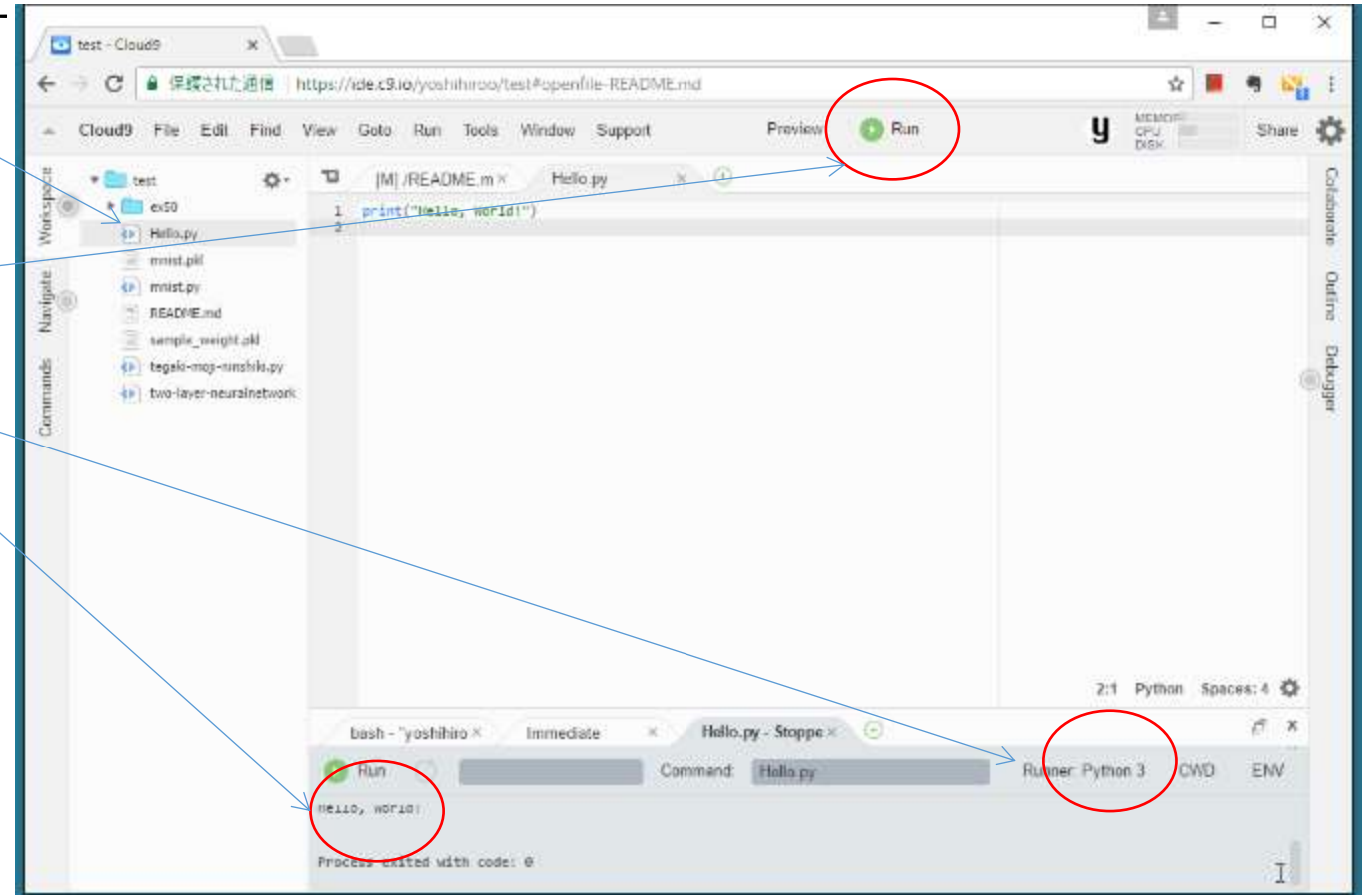
ワークスペースの作成時の名前がディレクトリ名になっています
この画面の場合「test」。



mnist.pklは重たいファイルなので時間がかかります。
ここに進行状況が表示されます。

Cloud9によるPython開発環境（6）

- ⑩ サンプルファイルの実行
 - Hello.pyをダブルクリック
 - ソースコードが表示される
 - 右上のRUNをクリック
 - 「Python 3」を確認
 - Hello, World!の表示を確認





蛇足

いろいろな呼び方

• 行列 = マトリックス = 二次元配列 = 二次元行列
数学っぽい 英語 プログラムのときよく使う 次元を明示的に

• ベクトル = 配列 = リスト
数学っぽい プログラム.. 特にPython

表記方法

- 行列は大文字 A, B, C..
- ベクトルは小文字 a, b, c..

Python上で行列計算を行う

```
import numpy as np
a = np.array([1, 2, 3])
print(a)
b = a + 3
print(b)
c = b * 10
print(c)
D = np.array( [[1,2,4], [3,5,7]] )
print(D)
E = np.array( [[3,4], [5,6], [5,4]] )
print(E)
```

```
1 import numpy as np
2 a = np.array([1, 2, 3])
3 print(a)
4 b = a + 3
5 print(b)
6 c = b * 10
7 print(c)
8 D = np.array( [[1,2,4], [3,5,7]] )
9 print(D)
10 E = np.array( [[3,4], [5,6], [5,4]] )
11 print(E)
12 |
```

bash - "yoshihiro ×

Immediate (Java ×



Run



Your code is running at <https://test-deep-learn.com/>

Important: use `os.getenv(PORT, 8080)` as the port

```
[1 2 3]
[4 5 6]
[40 50 60]
[[1 2 4]
 [3 5 7]]
[[3 4]
 [5 6]
 [5 4]]
```

Process exited with code: 0

先ほどの例をNumPyを使って計算してみましょう

$$\begin{pmatrix} 1 & 3 \\ 5 & 7 \\ 9 & 11 \end{pmatrix} \cdot \begin{pmatrix} 11 \\ 22 \end{pmatrix} = \begin{pmatrix} 77 \\ 209 \\ 341 \end{pmatrix}$$

先ほどの例をNumPyを使って計算してみましょう

$$\begin{pmatrix} 1 & 3 \\ 5 & 7 \\ 9 & 11 \end{pmatrix} \cdot \begin{pmatrix} 11 \\ 22 \end{pmatrix} = \begin{pmatrix} 77 \\ 209 \\ 341 \end{pmatrix}$$

X y z

```
import numpy as np
X = np.array( [[1,3], [5,7], [9,11]] )
y = np.array( [11,22] )
z = np.dot(X, y)
print(z)
```

← np.dotは内積を計算するコマンド

先ほどの例をNumPyを使って計算してみましょう

$$\underset{y}{(11 \ 22)} \cdot \underset{X}{\begin{pmatrix} 1 & 3 \\ 5 & 7 \\ 9 & 11 \end{pmatrix}} = \boxed{?}$$

```
import numpy as np
X = np.array( [[1,3], [5,7], [9,11]] )
y = np.array( [11,22] )
z = np.dot(y, X)
print(z)
```

← Xとyを入れ替えて内積計算したらどうなりますか？

その他のNumPyのコマンドをいくつか 試してみましょう

青文字箇所を追加

```
import numpy as np
X = np.array( [[1,3], [5,3], [9,11]] )
y = np.array( [11,22] )
z = np.dot(X, y)
print(z)
print(X[0])
print(X[0][1])
print(X.shape)
```

X[3]と見ようとしたらどうなるでしょうか？

同じく、X[0][3]は？

Xのところをyやzに変えたらどうなるでしょうか？

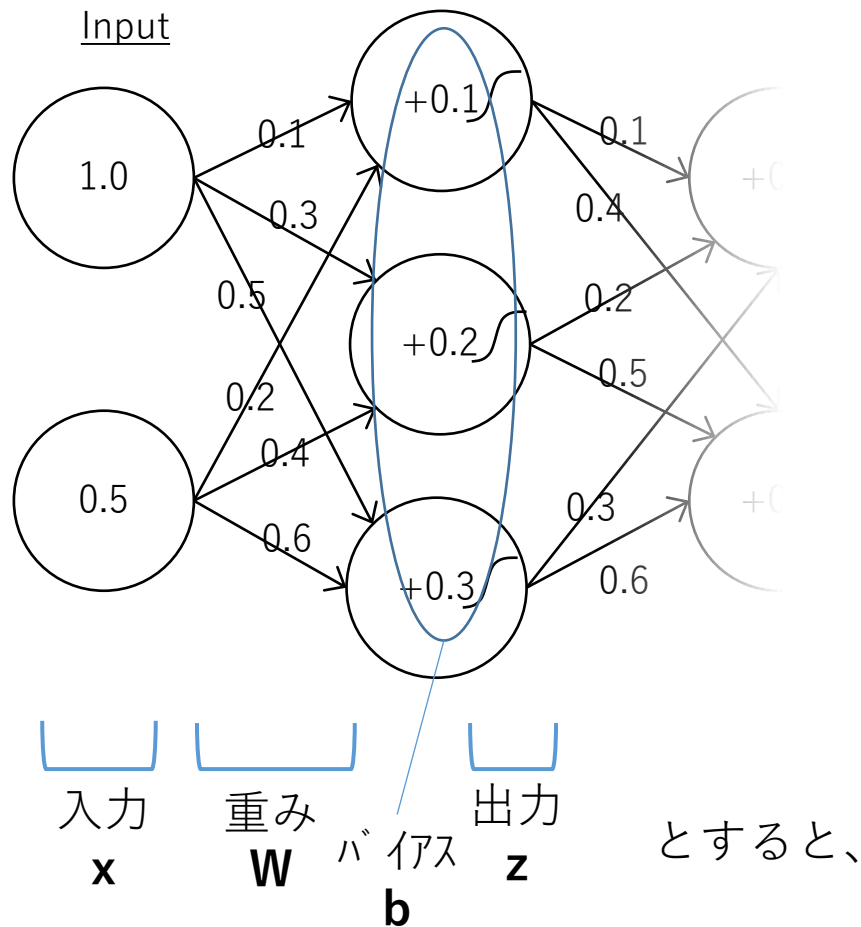
なぜ、行列（内積）なんていう道具を使うのか？

それは、行列を使うとニューラルネットワークをシンプルに記述できるから。

ニューラルネットワークを行列計算で表現してみる

sig() …シグモイド関数

先ほどの模型の前半部分



$$\begin{aligned} z &= \text{sig}(W \cdot x + b) \\ &= \text{sig}\left(\begin{pmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \\ 0.5 & 0.6 \end{pmatrix} \cdot \begin{pmatrix} 1.0 \\ 0.5 \end{pmatrix} + \begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix}\right) \\ &= \text{sig}\left(\begin{pmatrix} 0.2 \\ 0.5 \\ 0.8 \end{pmatrix} + \begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix}\right) \\ &= \text{sig}\left(\begin{pmatrix} 0.3 \\ 0.7 \\ 1.1 \end{pmatrix}\right) = \begin{pmatrix} 0.574 \\ 0.668 \\ 0.750 \end{pmatrix} \end{aligned}$$

ニューラルネットワークを行列計算で表現してみる

先ほどの模型の前半部分

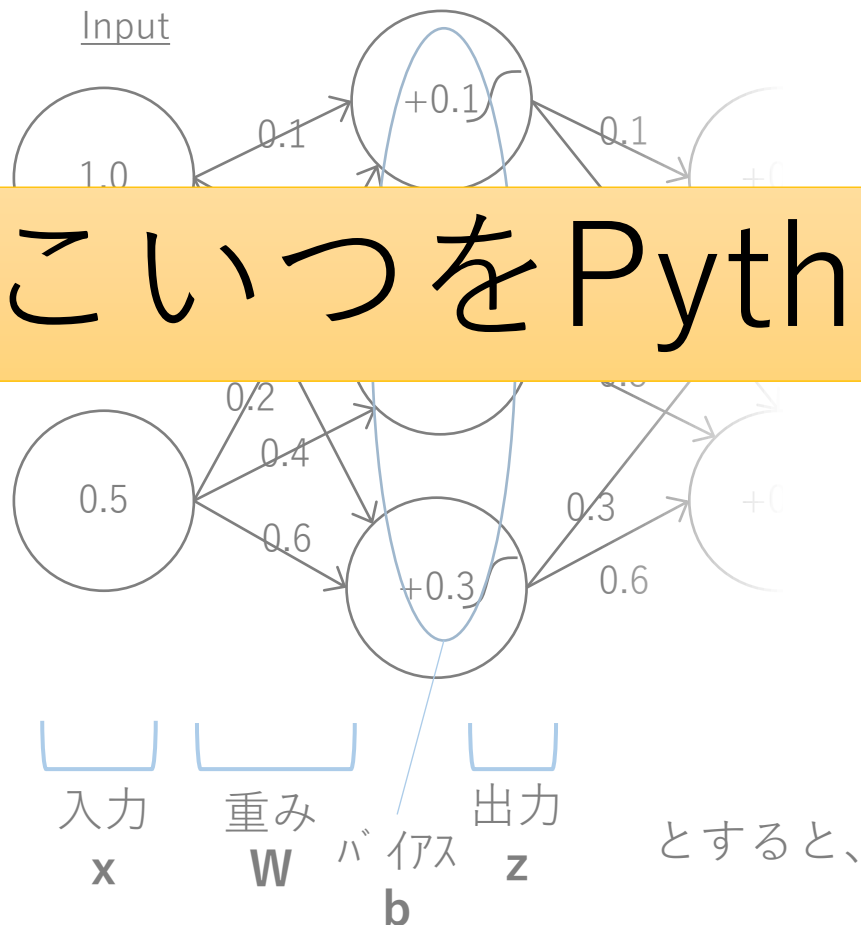
sig() …シグモイド関数

$$z = \text{sig}(W \cdot x + b)$$

こいつをPythonで実装してみよう！

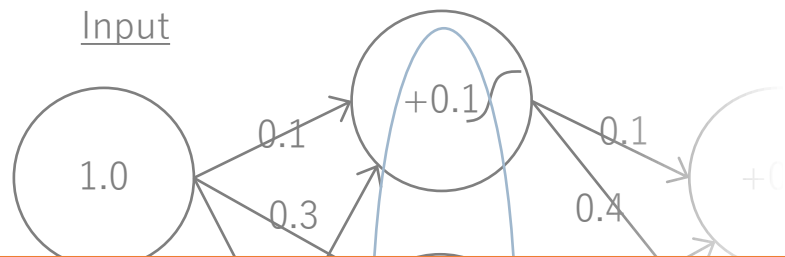
$$= \text{sig}\left(\begin{pmatrix} 0.2 \\ 0.5 \\ 0.8 \end{pmatrix} + \begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix}\right)$$

$$= \text{sig}\left(\begin{pmatrix} 0.3 \\ 0.7 \\ 1.1 \end{pmatrix}\right) = \begin{pmatrix} 0.574 \\ 0.668 \\ 0.750 \end{pmatrix}$$



ニューラルネットワークを行列計算で表現してみる

先ほどの模型の前半部分



sig() …シグモイド関数

$$\mathbf{z} = \text{sig}(\mathbf{W} \cdot \mathbf{x} + \mathbf{b})$$

$$= \text{sig}\left(\begin{pmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \\ 0.5 & 0.6 \end{pmatrix} \cdot \begin{pmatrix} 1.0 \\ 0.5 \end{pmatrix} + \begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix}\right)$$

$$\begin{pmatrix} 0.2 \\ 0.5 \\ 0.8 \end{pmatrix} + \begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix}$$

$$\begin{pmatrix} 0.3 \\ 0.7 \\ 1.1 \end{pmatrix} = \begin{pmatrix} 0.574 \\ 0.668 \\ 0.750 \end{pmatrix}$$

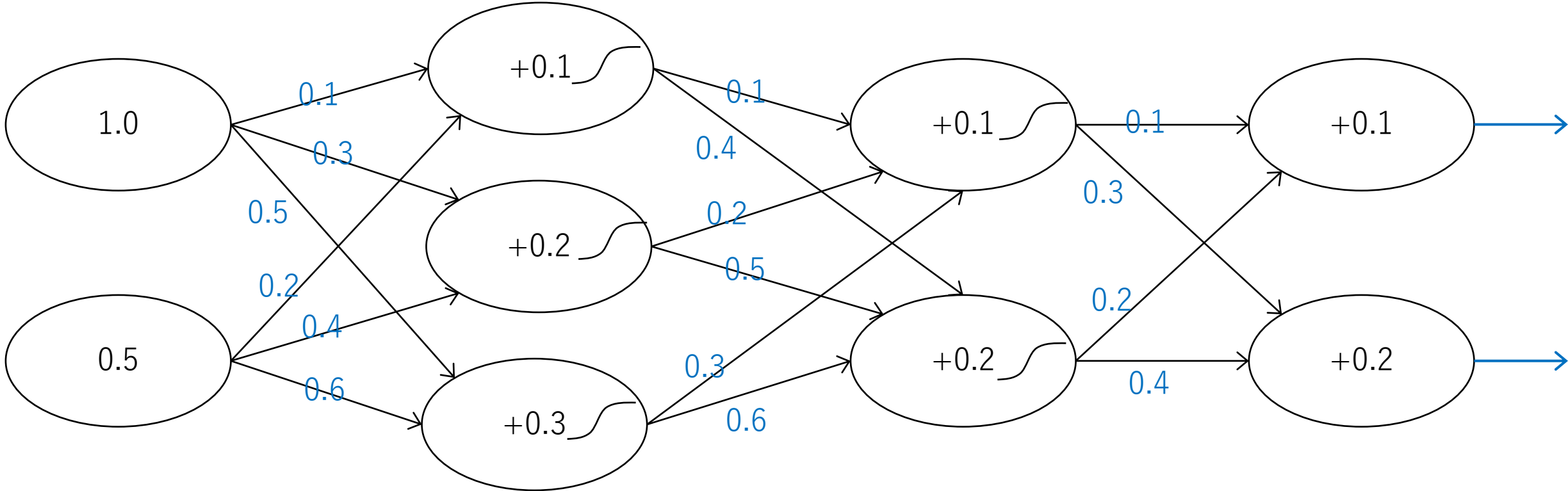
```
import numpy as np

def sig(x):
    return 1 / (1 + np.exp(-x))

W = np.array([[0.1, 0.2], [0.3, 0.4], [0.5, 0.6]])
x = np.array([1.0, 0.5])
b = np.array([0.1, 0.2, 0.3])

z = sig( np.dot(W, x) + b )
print(z)
```

先ほどのニューラルネットの例



x	W1	b1	z1	W2	b2	z2	W3	b3	z3
$\begin{pmatrix} 1.0 \\ 0.5 \end{pmatrix}$	$\begin{pmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \\ 0.5 & 0.6 \end{pmatrix}$	$\begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix}$	$\begin{pmatrix} ? \\ ? \\ ? \end{pmatrix}$	$\begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \end{pmatrix}$	$\begin{pmatrix} 0.1 \\ 0.2 \end{pmatrix}$	$\begin{pmatrix} ? \\ ? \end{pmatrix}$	$\begin{pmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{pmatrix}$	$\begin{pmatrix} 0.1 \\ 0.2 \end{pmatrix}$	$\begin{pmatrix} ? \\ ? \end{pmatrix}$

$z1 = \text{sig}(W1 \cdot x + b1)$

$z2 = \text{sig}(W2 \cdot z1 + b2)$

$z3 = W3 \cdot z2 + b3$

sig() …シグモイド関数

```
import numpy as np

def sig(x):
    return 1 / (1 + np.exp(-x))

W1 = np.array([[0.1,0.2], [0.3,0.4], [0.5,0.6]])
x = np.array([1.0, 0.5])
b1 = np.array([0.1, 0.2, 0.3])
z1 = sig( np.dot(W1, x) + b1 )

W2 = np.array([[0.1,0.2,0.3], [0.4,0.5,0.6]])
b2 = np.array([0.1, 0.2])
z2 = sig( np.dot(W2, z1) + b2 )

W3 = np.array([[0.1,0.2], [0.3,0.4]])
b3 = np.array([0.1, 0.2])
z3 = np.dot(W3, z2) + b3

print(z3)
```

コンピューターにニューラルネットワークの演算をさせる方法を手に入れた！

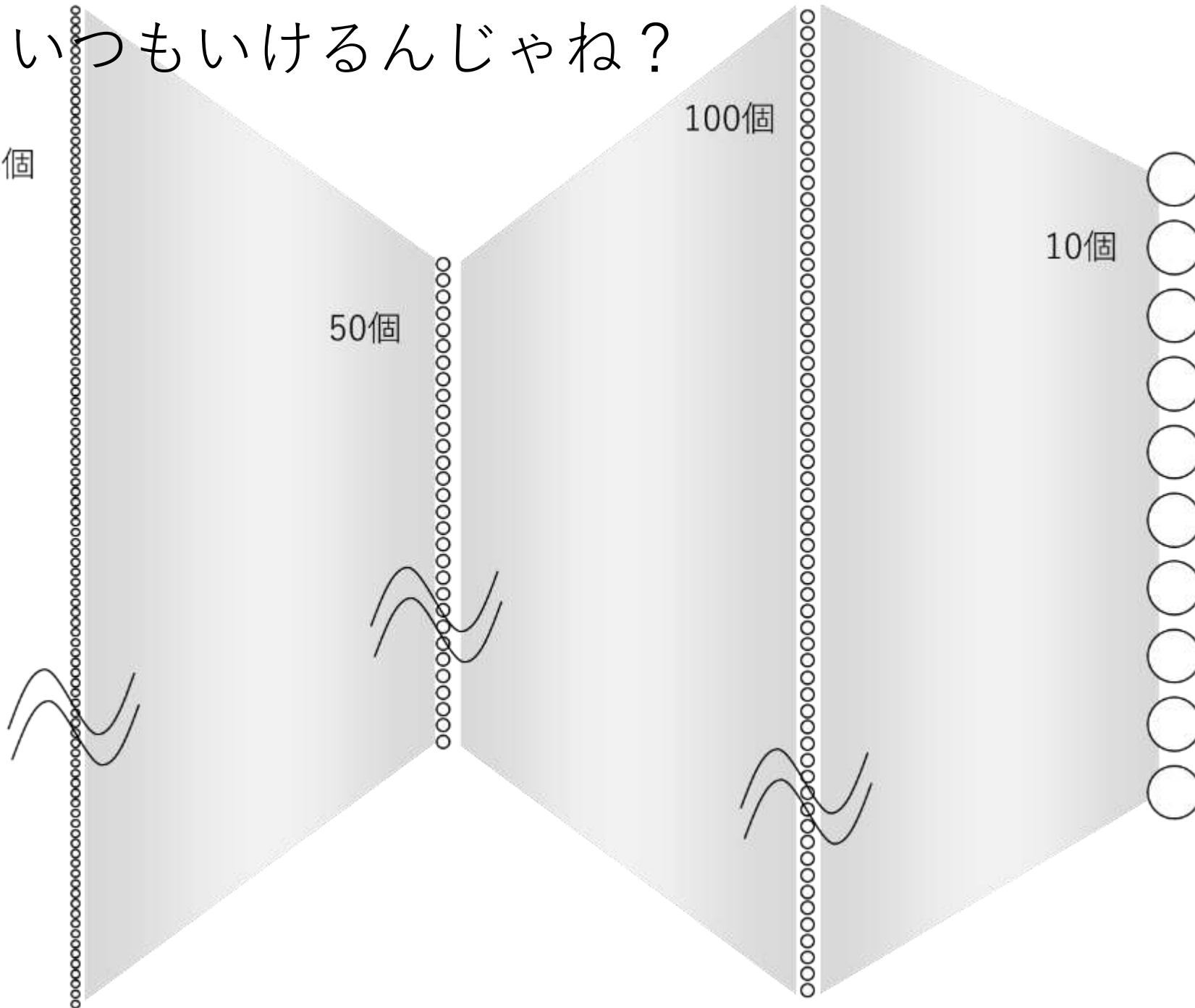
だったらこいつもいけるんじゃない？

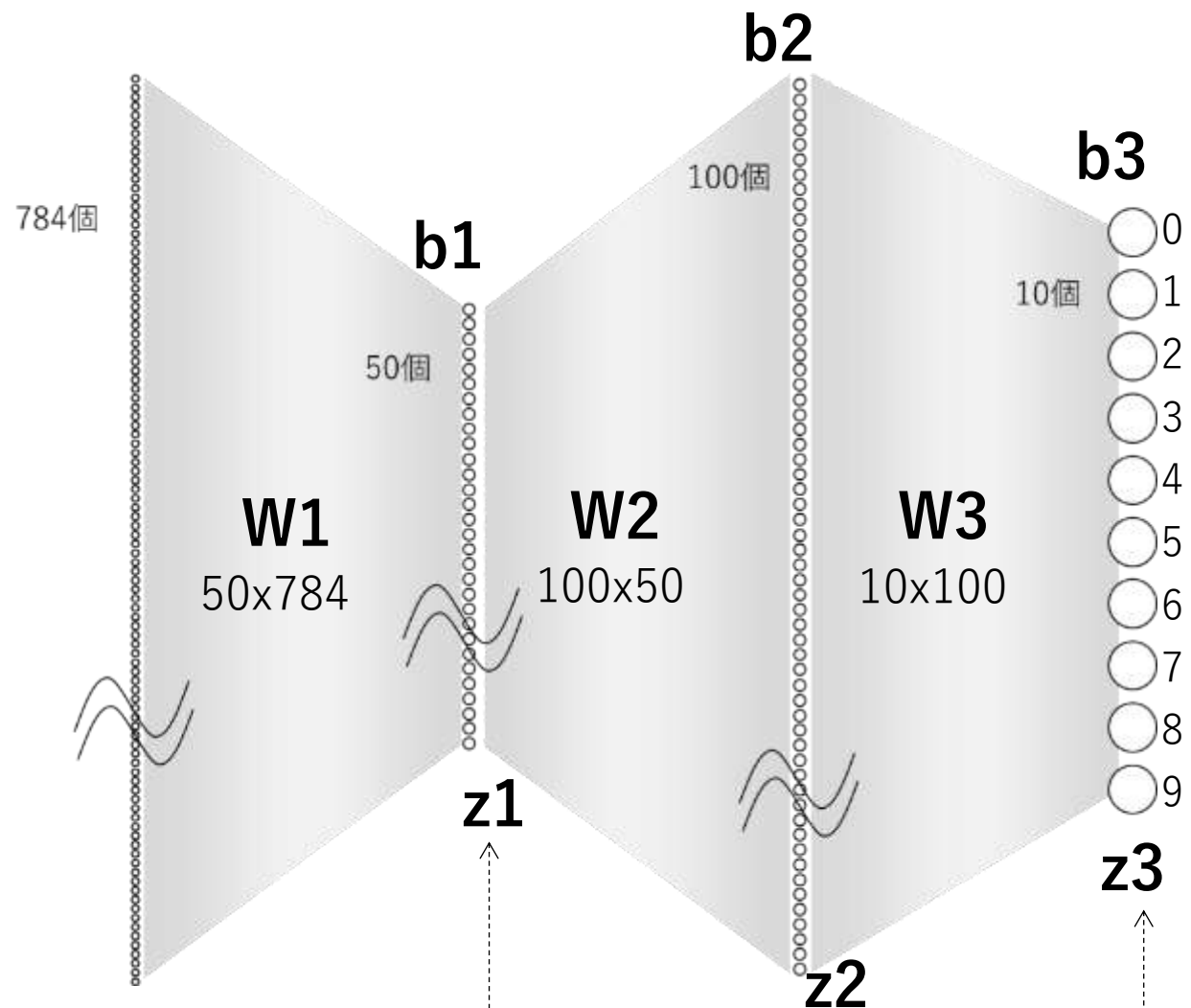
784個

50個

100個

10個





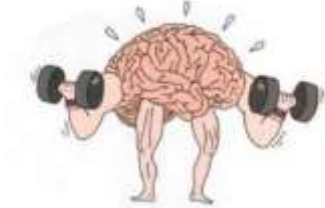
$$z1 = \text{sig}(W1 \cdot x + b1)$$

$$z2 = \text{sig}(W2 \cdot z1 + b2)$$

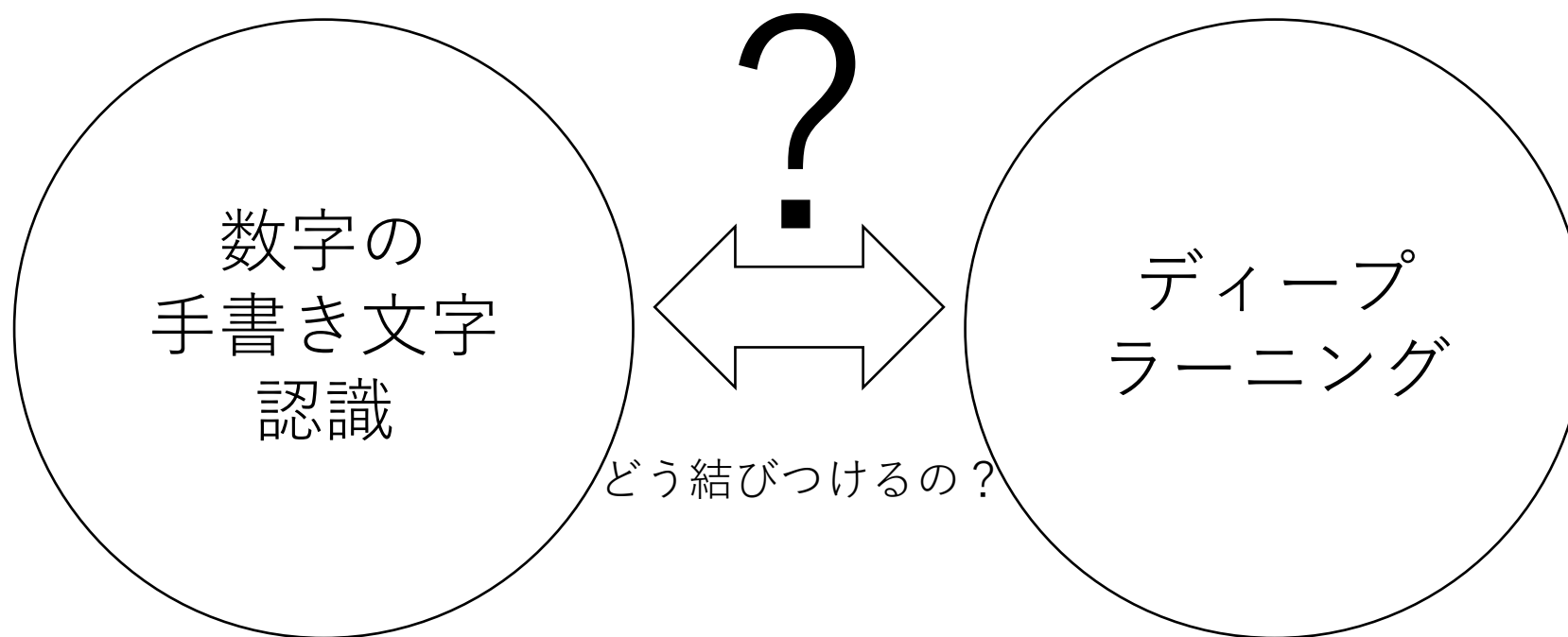
$$z3 = W3 \cdot z2 + b3$$

ここの理解、
今日の踏ん張りどころです

脳みそに汗をかけ



数字の手書き文字認識をさせてみる



どんなデータを使うのか

<http://yann.lecun.com/exdb/mnist/>

THE MNIST DATABASE

of handwritten digits

[Yann LeCun](#), Courant Institute, NYU

[Corinna Cortes](#), Google Labs, New York

[Christopher J.C. Burges](#), Microsoft Research, Redmond

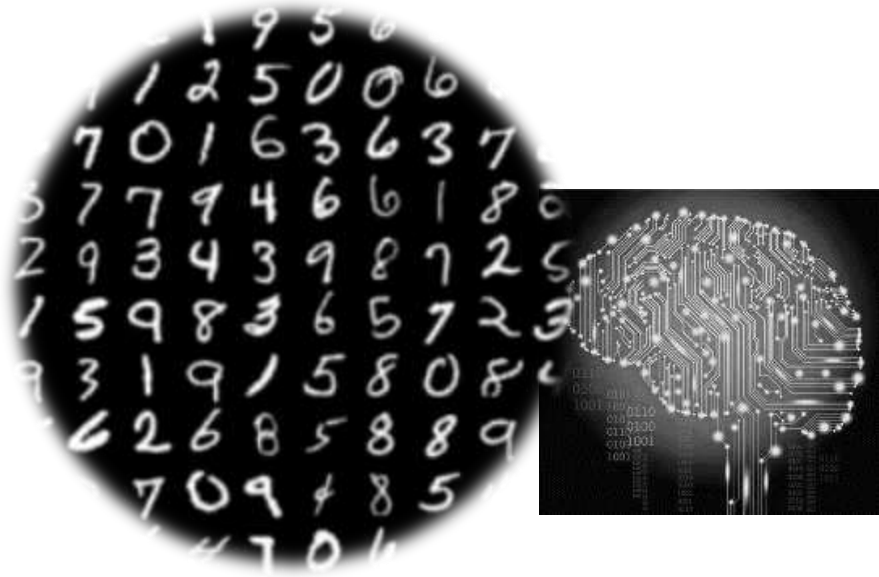
The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition



ディープラーニングによる文字認識には、大きく二つのフェーズがある

訓練用データを使って
学習させる

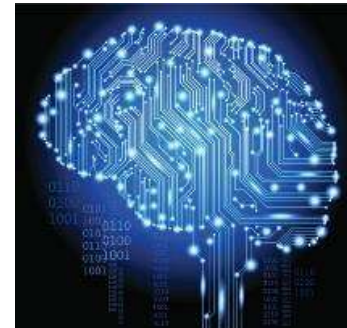


汎用データを判別させる
(訓練用データとは別)

5

7

9



これは
「5」
です

これは
「7」
です

これは
「9」
です

今日は6万枚の訓練用データと、1万枚の汎用データを用います

28



784個の数字の配列で、一つの
手書き文字を表現できる

そして、先ほどのこれ↓にほうりこむ

b2

この784という数字は、
 $28 \times 28 = 784$ 個の格子の
数に対応しています

784個

b1

50個

W1

50x784

W2

100x50

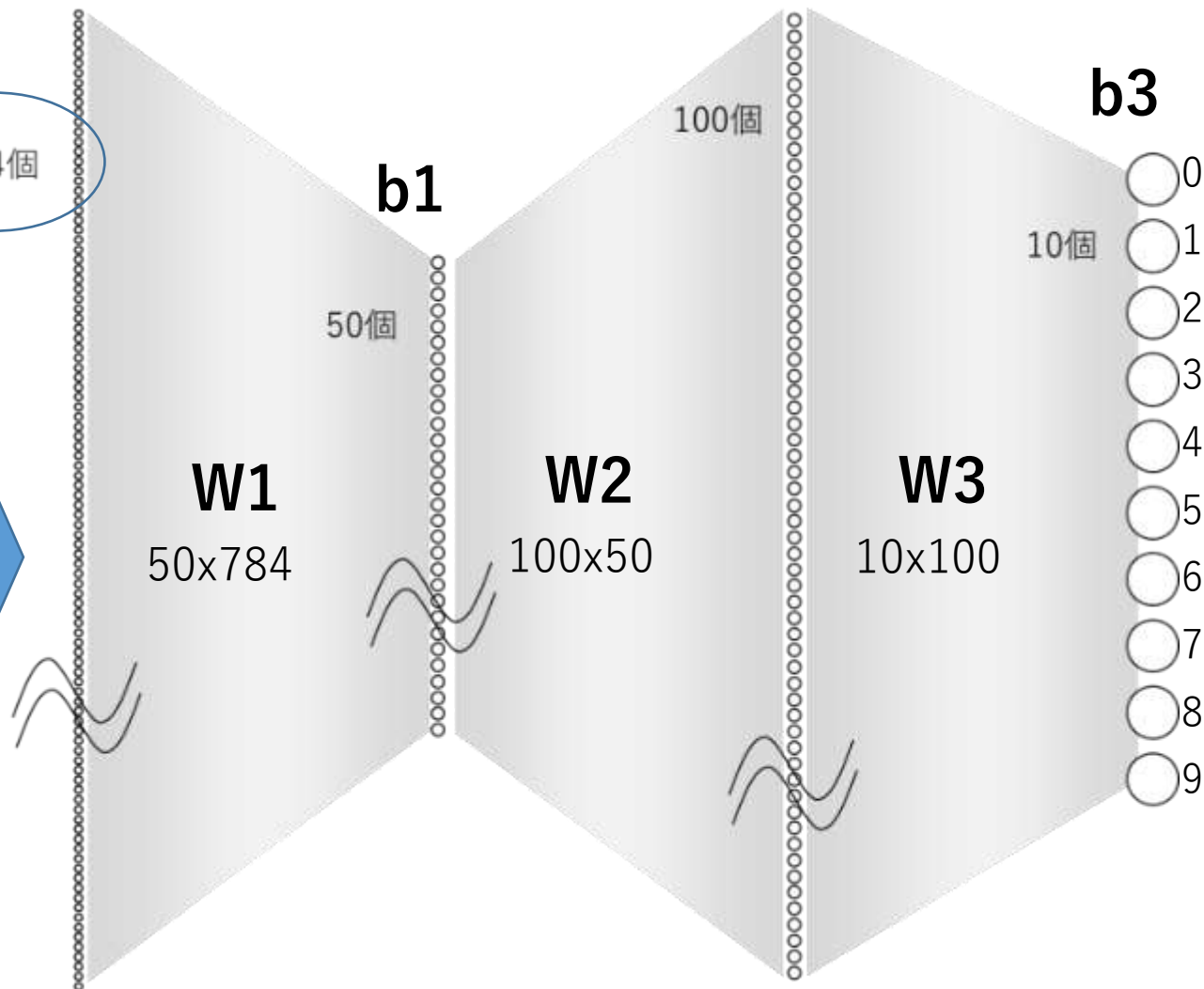
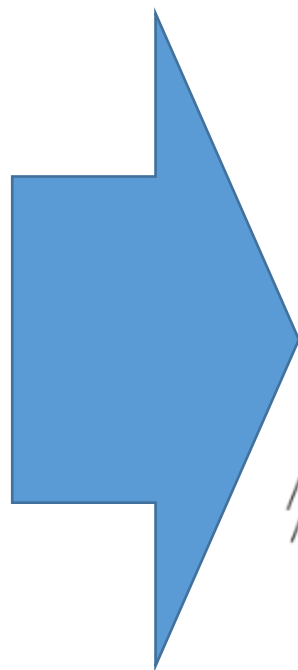
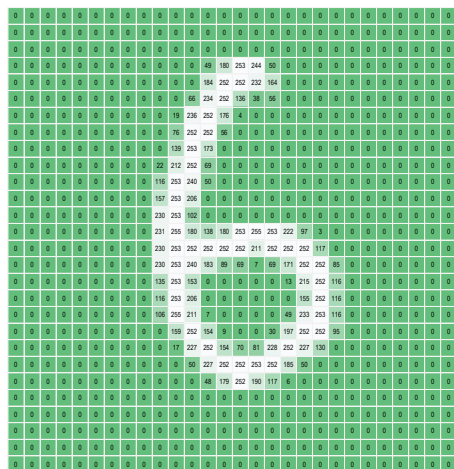
W3

10x100

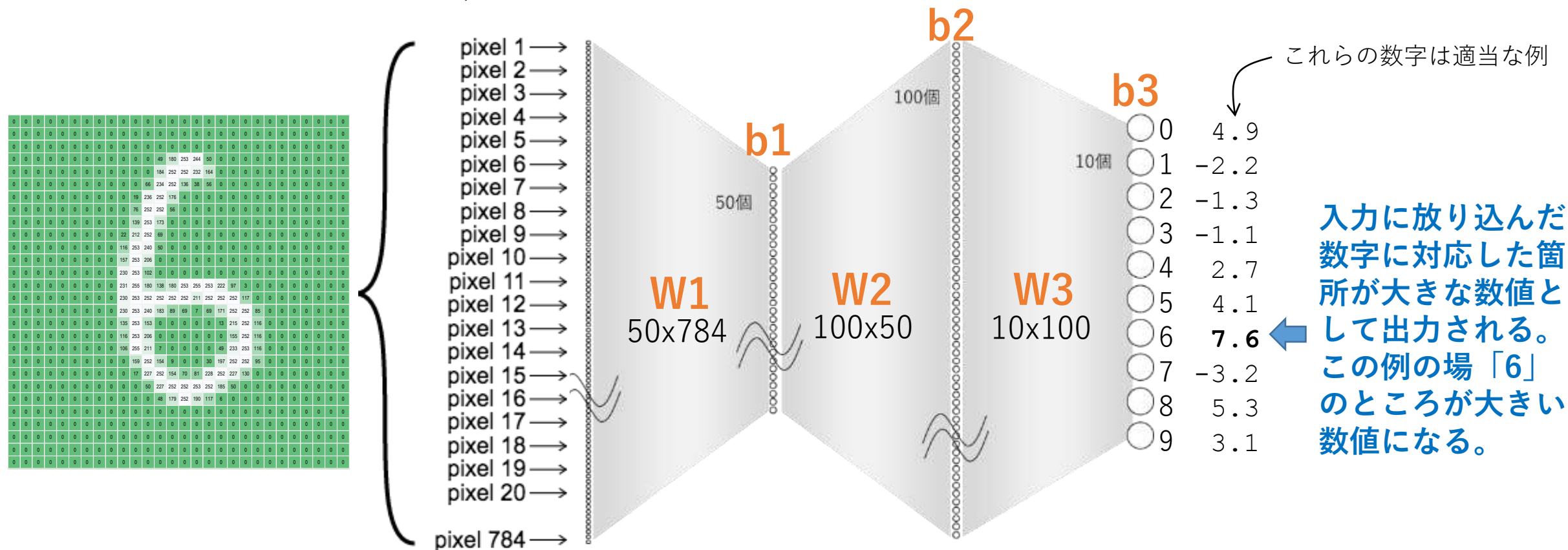
b3

10個

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9



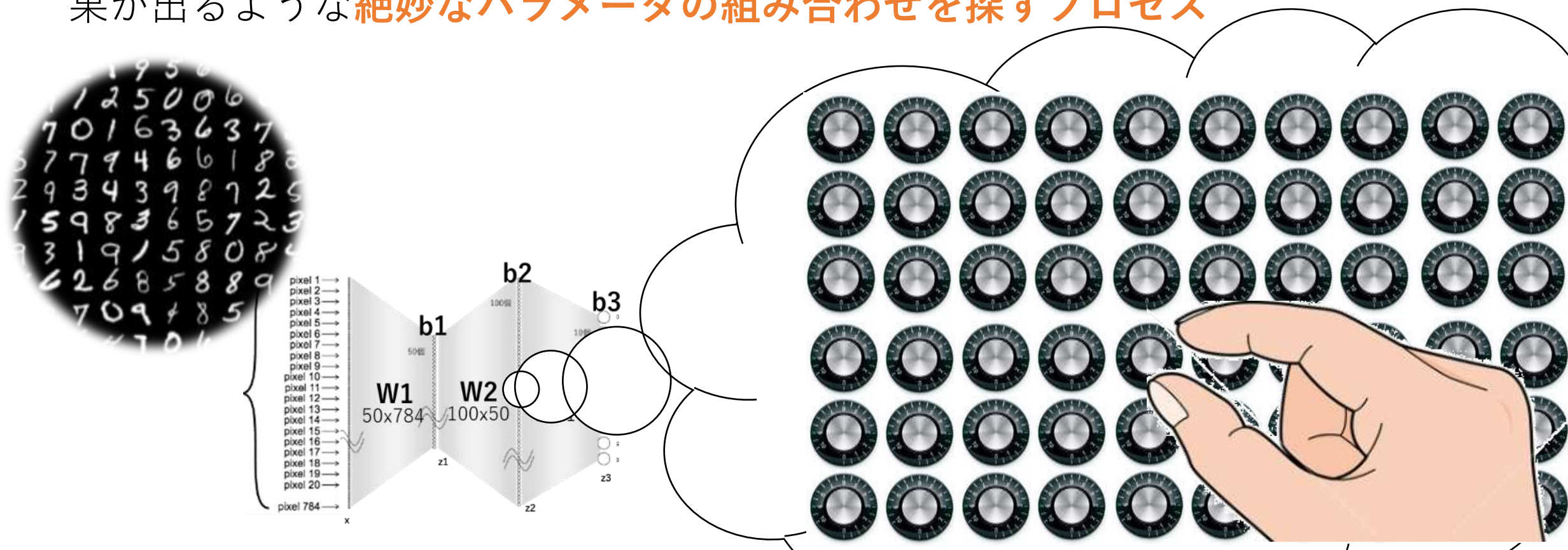
結果を先に言っておきますと、ディープラーニングによる文字認識とは、こういうことができちゃいます



W1, W2, W3, b1, b2, b3 (全部で45,350個の数字)それぞれに
“絶妙な組み合わせの数字”を置いたら、これができてしまう！
これが、ディープな(多層構造の)ニューラルネットワークの威力

ディープラーニングにおける学習とは

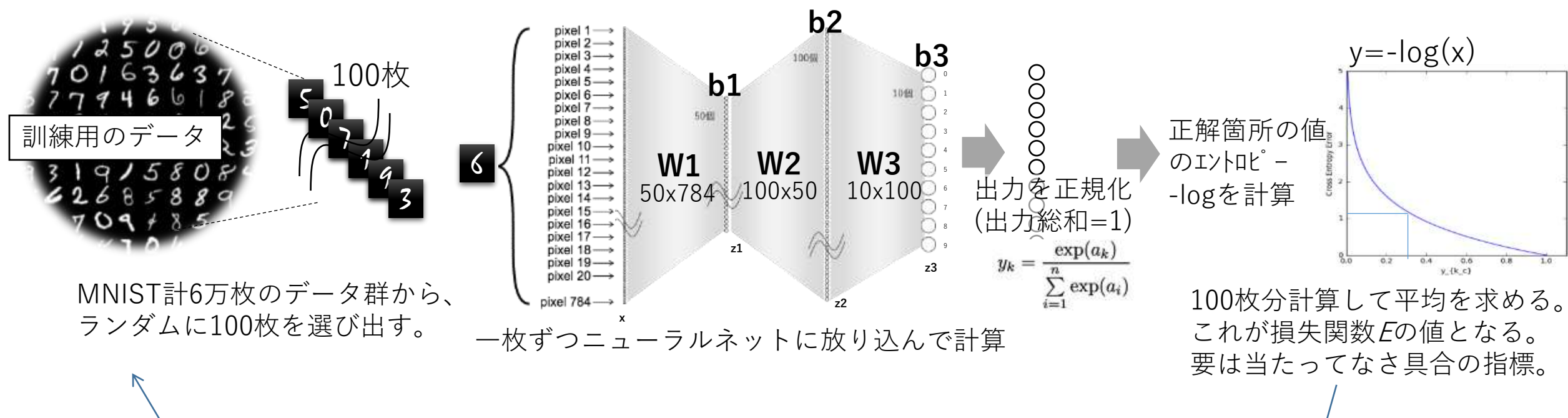
多層構造のニューラルネットワークに対して、大量の訓練用データを食べさせながら、コンピューターの計算力を頼りに、ちょっとずつパラメータ $W1$, $W2$, $W3$, $b1$, $b2$, $b3$ (全部で45,350個の数字)を変えながら正しい結果が出るような**絶妙なパラメータの組み合わせを探すプロセス**



学習のプロセスをもう少し詳しく見る

損失関数～当たってなさ具合を指標化

大量の訓練用データを食わせながら、コンピュータの計算力を頼りに、ちょっとずつパラメータ $W1, W2, W3, b1, b2, b3$ (全部で45,350個の数字)を変えながら正しい結果が出るような**絶妙なパラメータの組み合わせを探すプロセス**

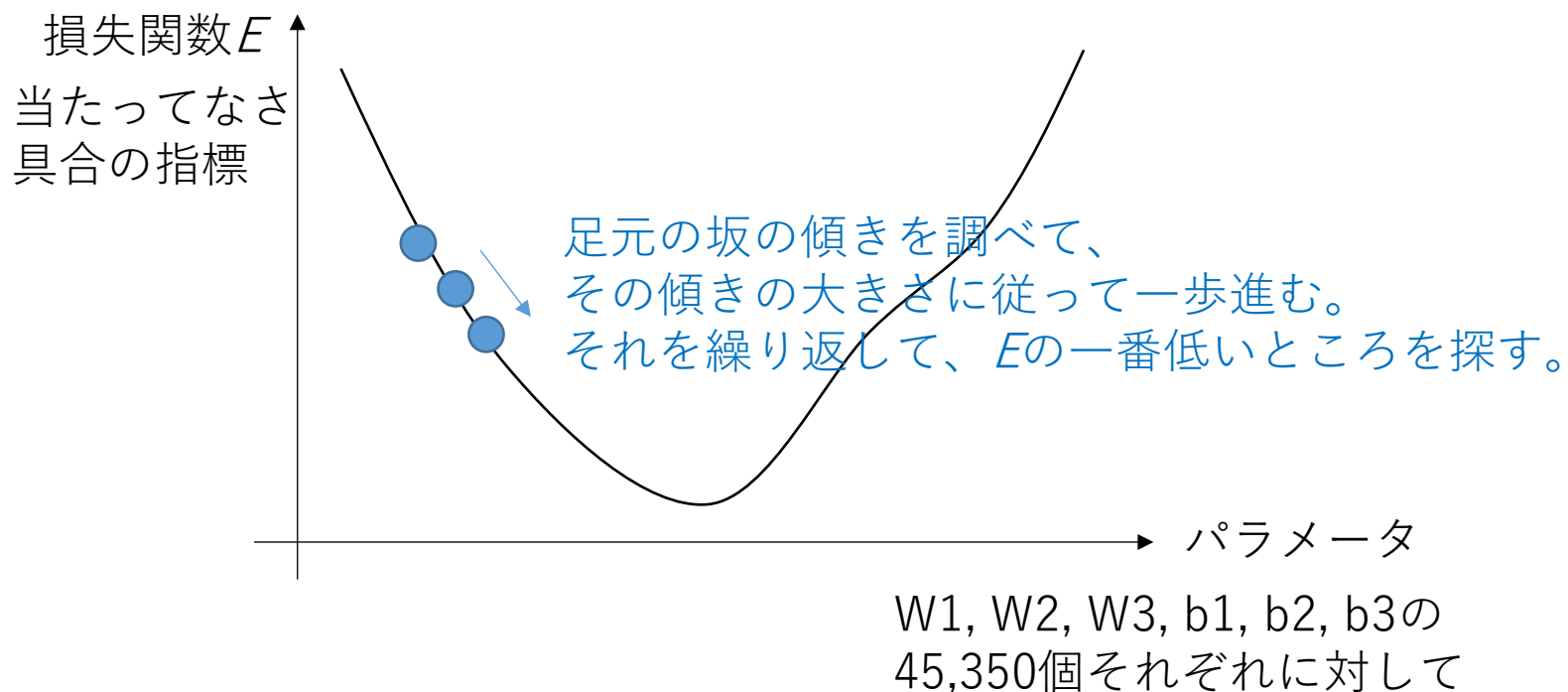


パラメータを少しずつ変えて繰り返し。 E が十分に小さくなるパラメータを探す。

学習のプロセスをもう少し詳しく見る

損失関数が小さくなる場所を探す

大量の訓練用データを食わせながら、コンピュータの計算力を頼りに、ちよつとずつパラメータ $W1, W2, W3, b1, b2, b3$ (全部で45,350個の数字)を変えながら正しい結果が出るような**絶妙なパラメータの組み合わせを探すプロセス**

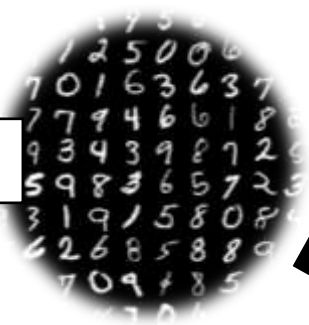


じゃ、学習って具体的にどうプログラムするの？

- 今日では時間の都合で学習のプログラミング実装は割愛
m()m
- 学習(コンピューターの数値計算)を効率よくさせるための手法はそれ自体が奥深い研究テーマであり、誤差逆伝搬法(バックプロパゲーション)、SGD、Momentum、AdaGrad、Adam、…など、専門用語がバンバン出てくる領域。今日はそのあたりの深入りはやめときます

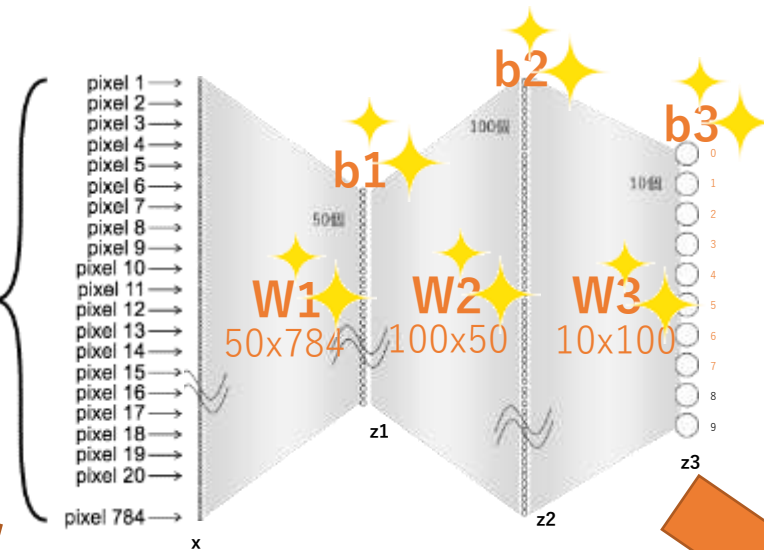
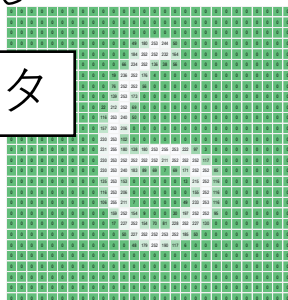
今日は学習の実装は割愛なので、まるで三分間クッキングのように、そこそこ絶妙なパラメータを手に入れたとする😊

訓練用のデータ



例えば「6」

テスト用のデータ

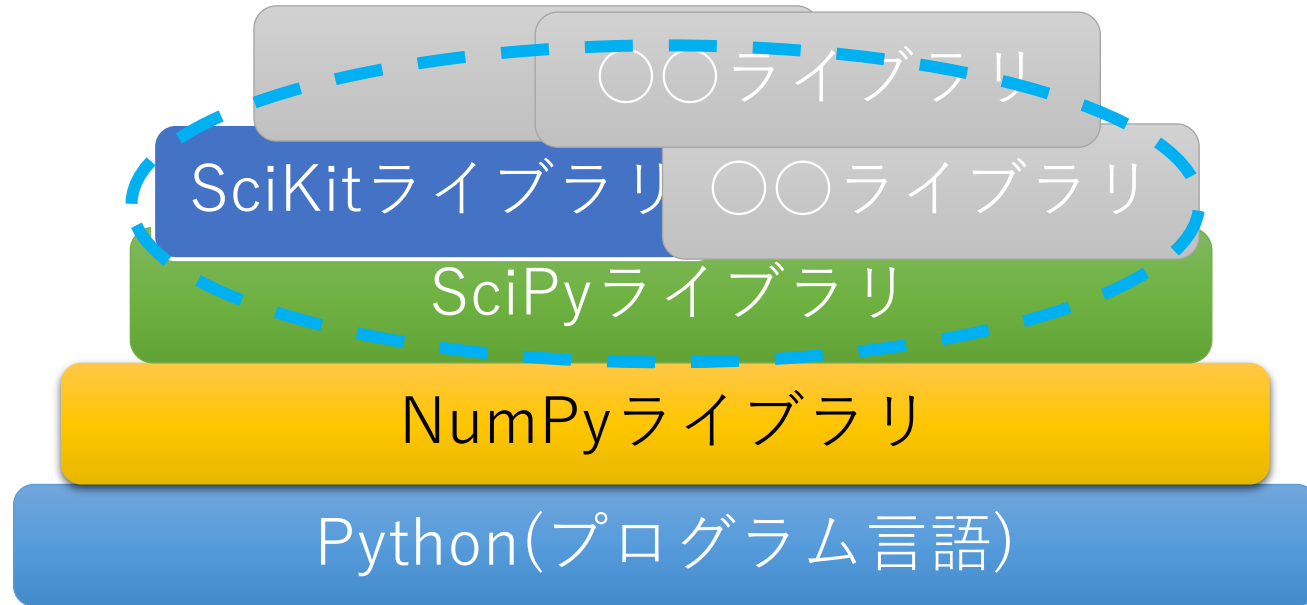


「6」と認識できるか？
実際にPythonで動かして
試してみよう！

ソースコード
tegaki-moji-ninshiki.py

ということで、一通りのおさらい

(ちょっと脱線) ディープラーニングでなぜPythonなのか？



Pythonにはディープラーニングのみならず、科学技術分野に有用なライブラリが豊富。
その土台を支えるのが、行列に関する演算を高速に・手軽に行うためのNumPyライブラリ。

便利なコマンド・関数をパッケージにして、広く他の人にも使えるようにしたもの

キラーアプリとしてNumPyの存在が大きい

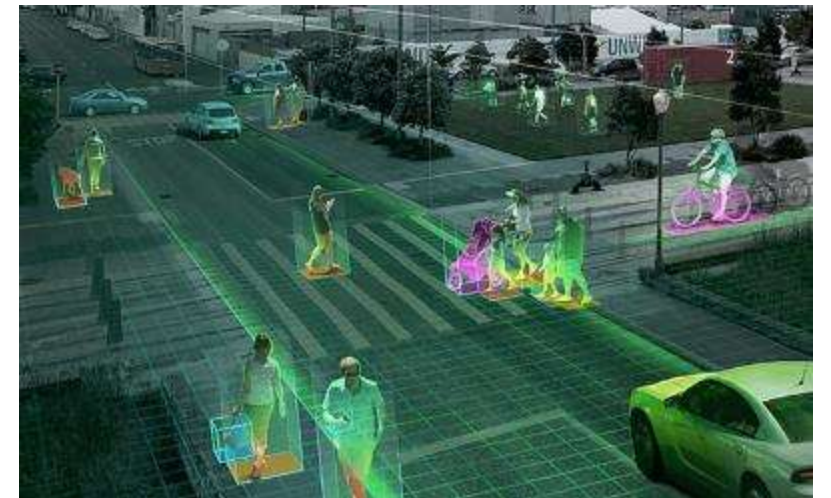
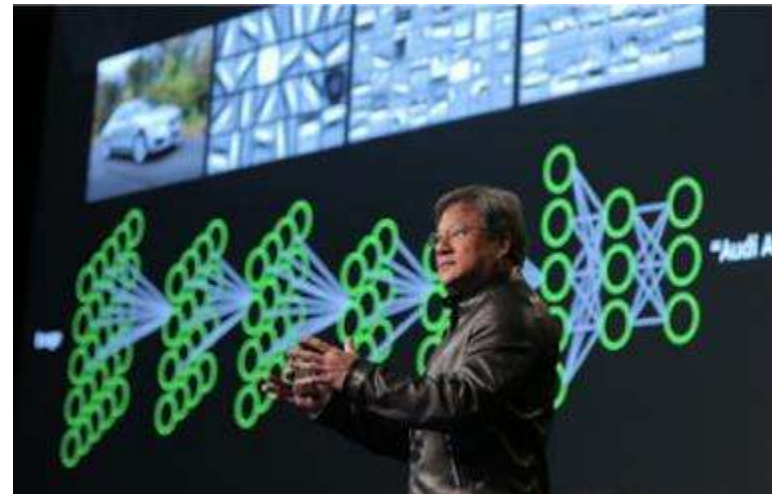
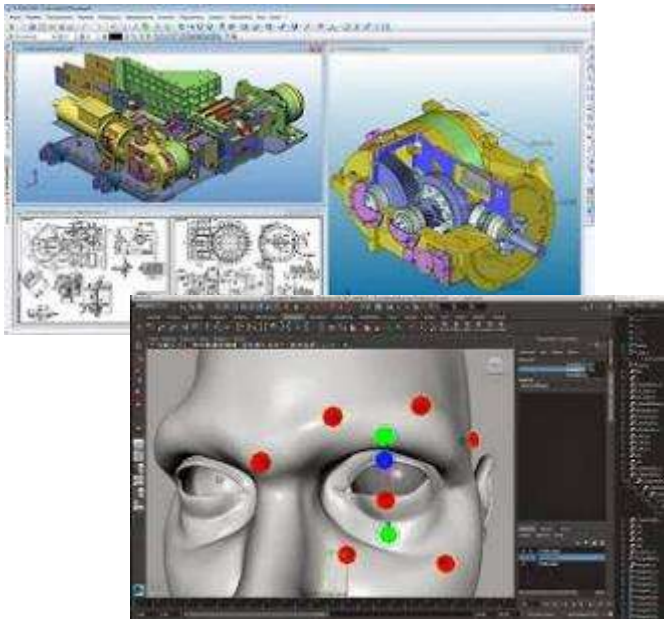
“謎のAI半導体メーカー”☺

話題の企業 – エヌビディア社



行列計算を超高速に並列処理できるチップを開発。それを組み込んだ各種ハードウェアと、それらを活用するためのソフトウェア群を提供。

CADやCG、ゲームなどの3Dグラフィックス領域から、ディープラーニングへ適応領域を拡大。いずれも膨大な行列計算を必要とするアプリケーション領域。



まだ潜り始めたばかり

まだまだ先は深いけど

- 今日の話のさらに続きとして、畳み込みニューラルネットワークへの発展、効率よく学習させるための様々なテクニックなど、学ぶべきテーマはまだまだあります
- しかし、その基本となるのは今日学んだニューラルネットワークと、それを多層構造化したモデルです
- これからニュースや記事でディープラーニングを見かけたときに、いままでよりも多少なりとも中身に親近感を持って理解できる一助となれば幸いです



手書き文字認識

http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html