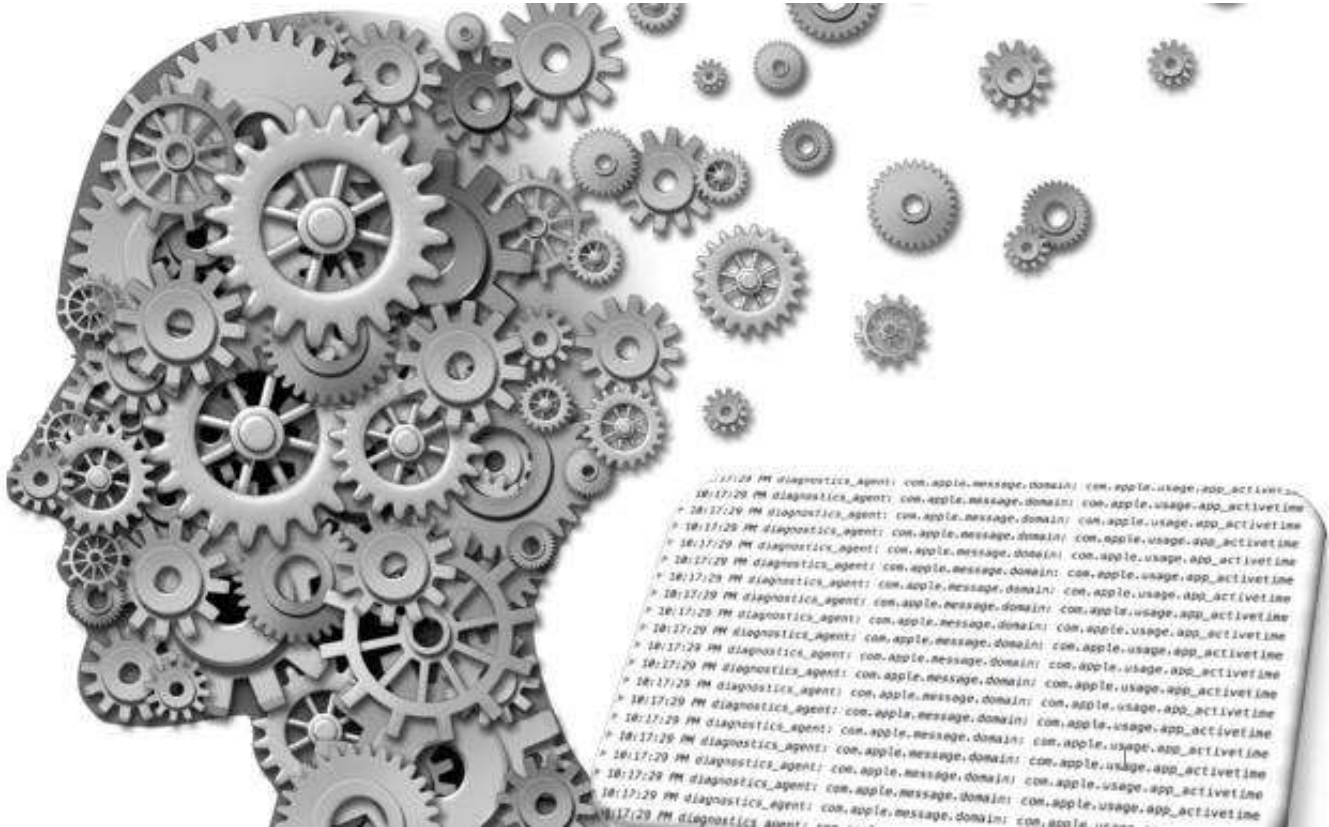
テクノベート勉強会

人工知能/ディープラーニングのプログラミング・ワークショップ



2017年12月2日

名古屋校 2011期 越智 由浩

今日のアジェンダ

- 今日の立ち位置、言葉の整理

- ディープラーニングの
  - 計算のやり方
  - 計算の道具

- ディープラーニングを用いた手書き文字(数字)認識の実例

今日のアジェンダ

- 今日の立ち位置、言葉の整理

- ディープラーニングの
  - 計算のやり方
  - 計算の道具

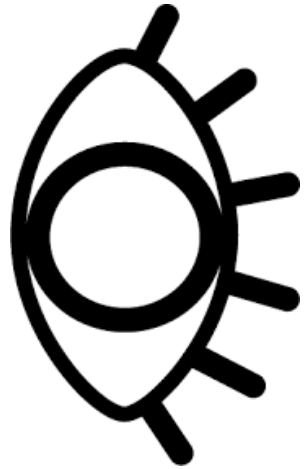
- ディープラーニングを用いた手書き文字(数字)認識の実例

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

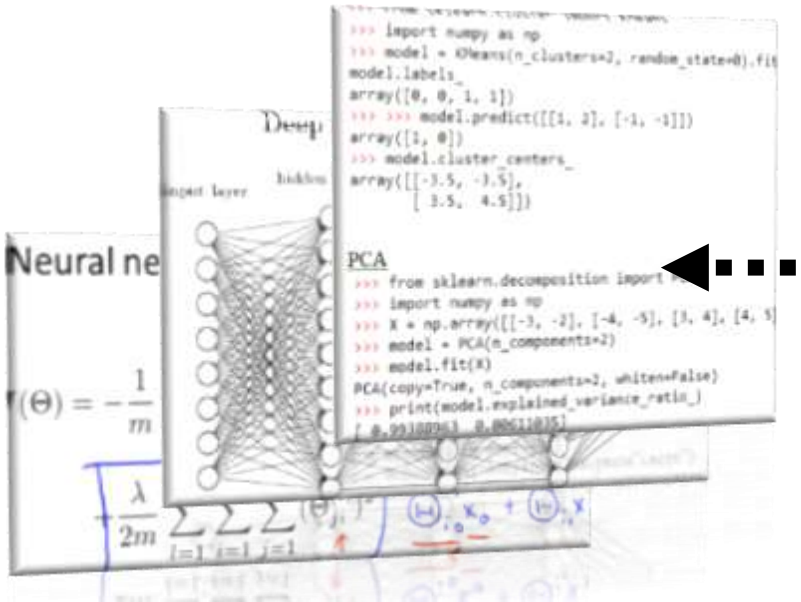
コトバノチガイ

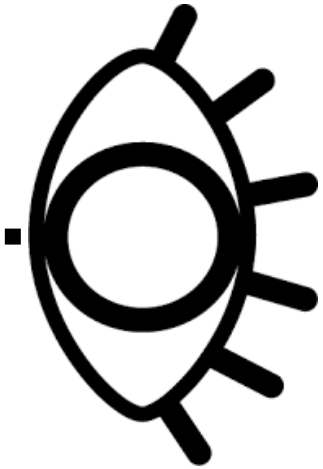


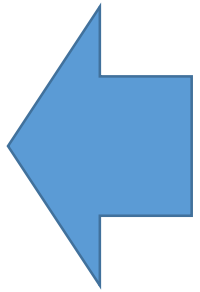
両方の目を持とう!

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



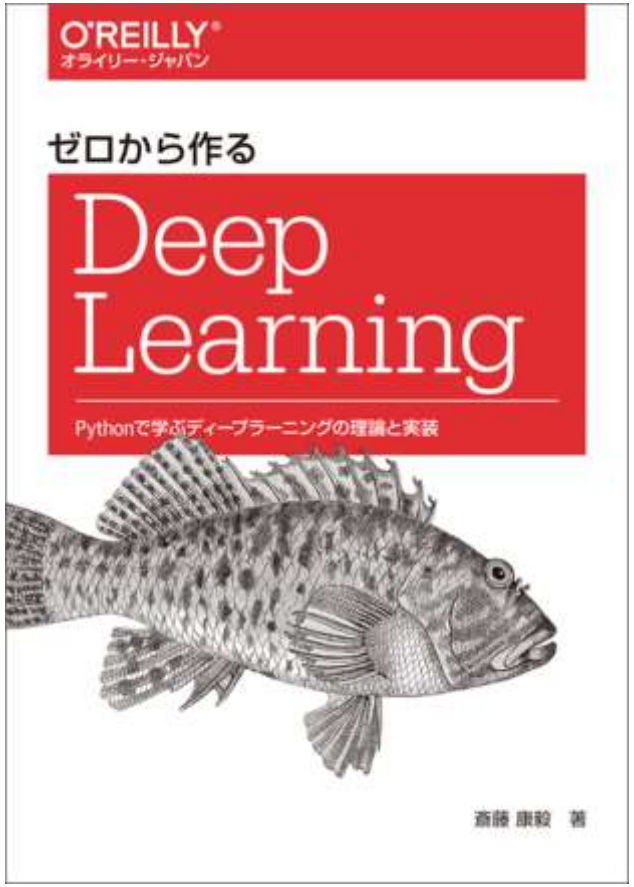




今日はこちら側の基礎的なところ

今日の内容のベースとなる本

- 内容、ソースコード、図など、こちらの本から引用しております
- Amazonでディープラーニングで検索するとベストセラー本として簡単に見つかります
- 復習兼ねてぜひご購読をお勧めします



3時間後の皆さんの状態（期待値）

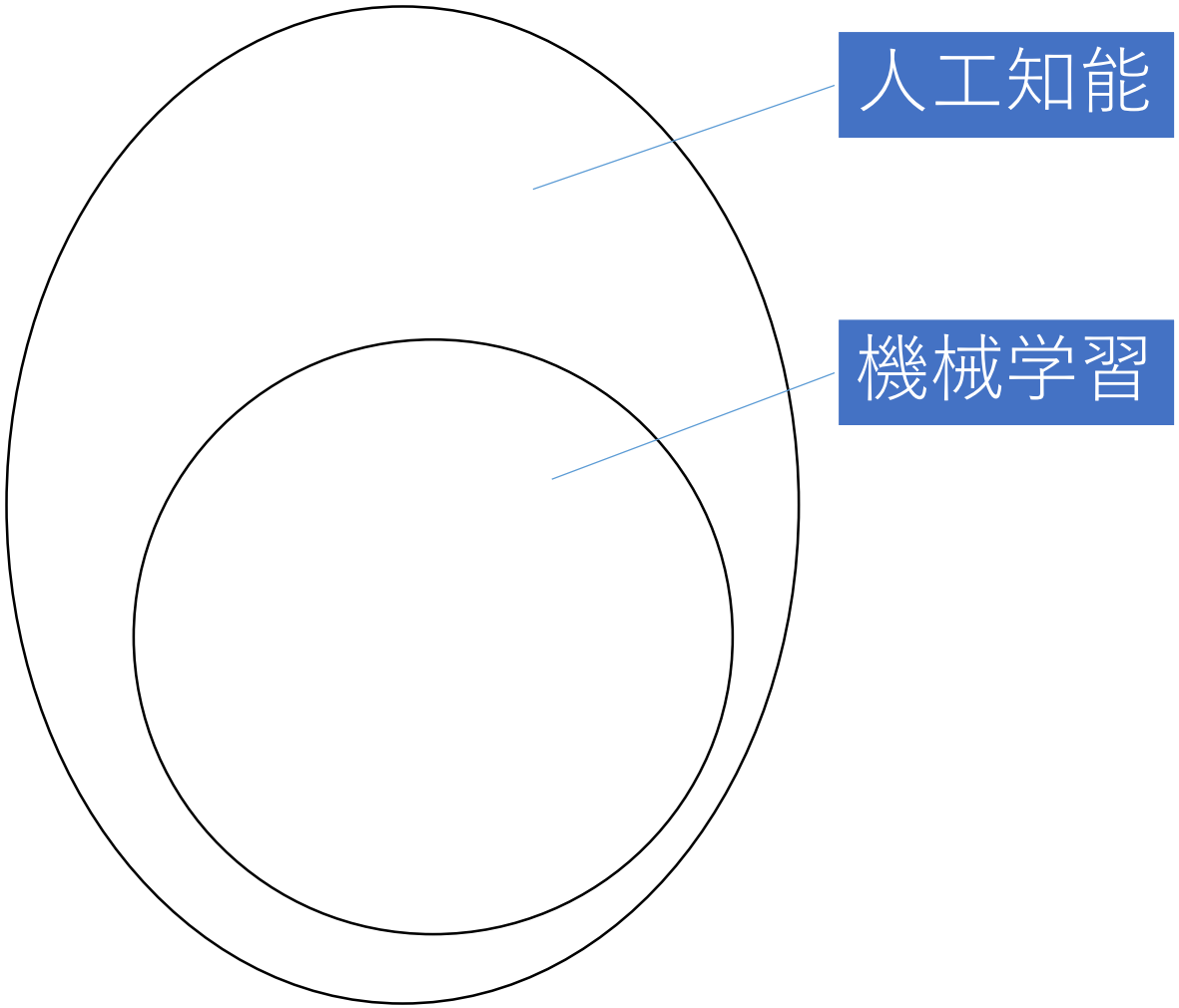
- さらに学ぼうとして、その手の本とかwebコンテンツを見たときに「あぁぁあぁ、なんとなく見たことあるぞ」という親近感を得られる
- Jupyterを触れるようになる―ディープラーニングを動かせる/試せる環境を手に入れることができる

まず、言葉の整理

人工知能

「お、こいつ賢いな!」と思わせるもの、ふるまい。
またそれを探求する学問領域。

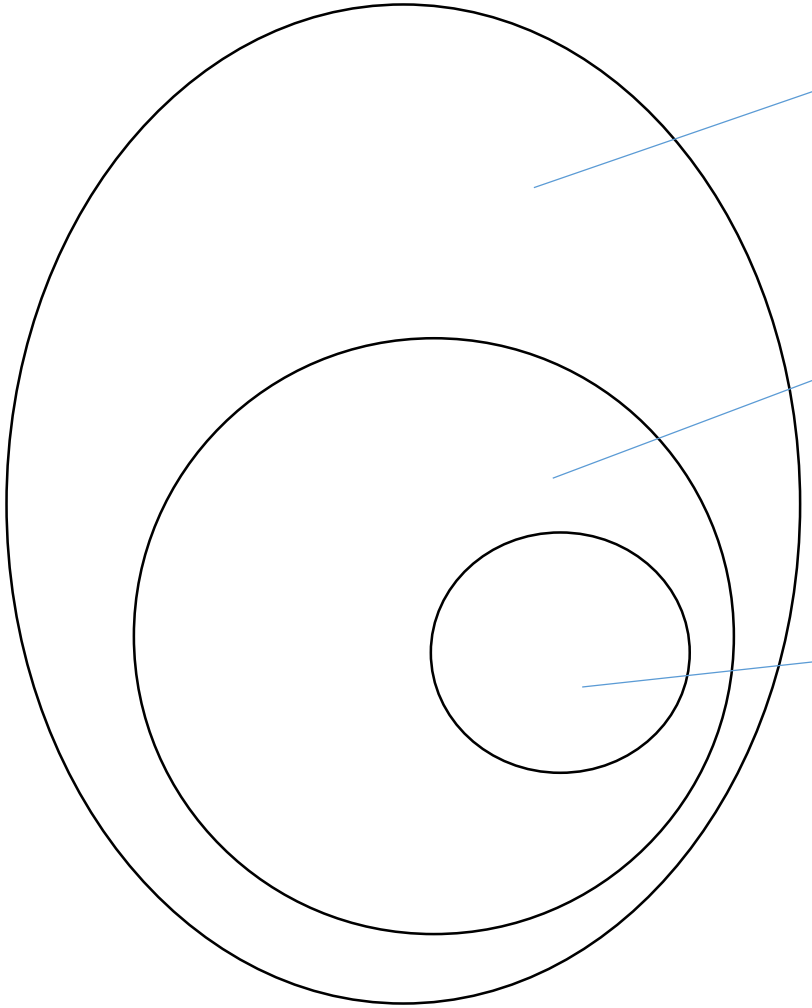
まず、言葉の整理



人工知能を実現する一つの手段。
過去のデータ(知見/経験)に基づいて：
- 将来を予測する
- 未知のものを分類する

例)
eコマースサイトのリコメンド
迷惑メールの自動振り分け

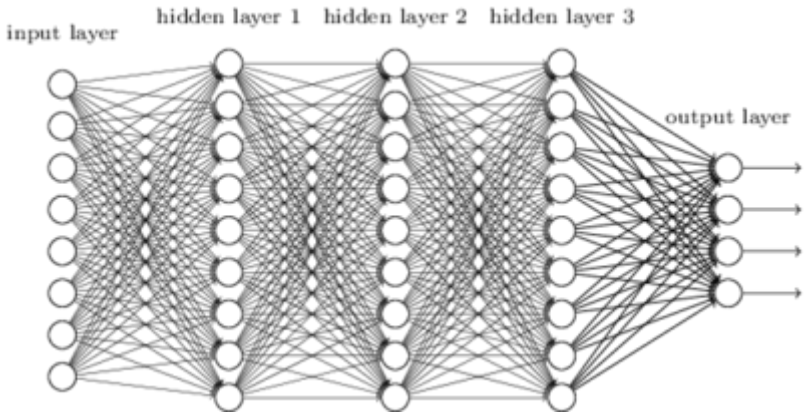
まず、言葉の整理



人工知能

機械学習

ディープラーニング



多階層(ディープな)ニューラルネットワークを用いた機械学習(ラーニング)

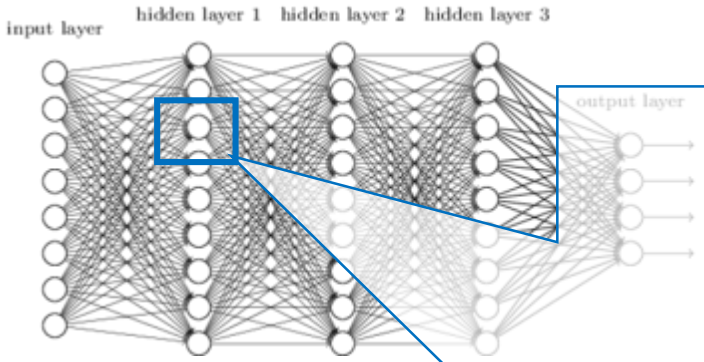
今日のアジェンダ

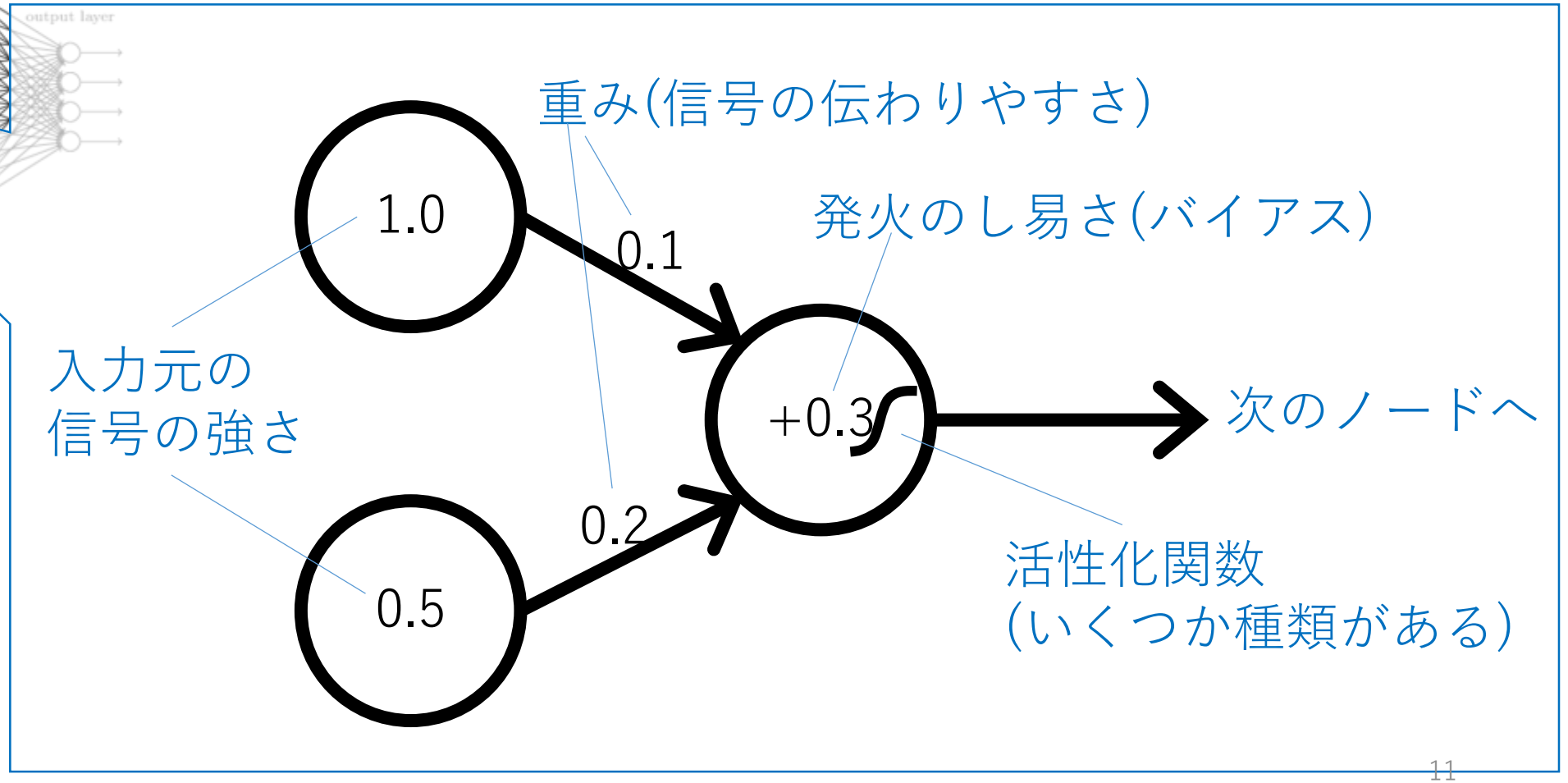
- 今日の立ち位置、言葉の整理

- ディープラーニングの
  - 計算のやり方
  - 計算の道具

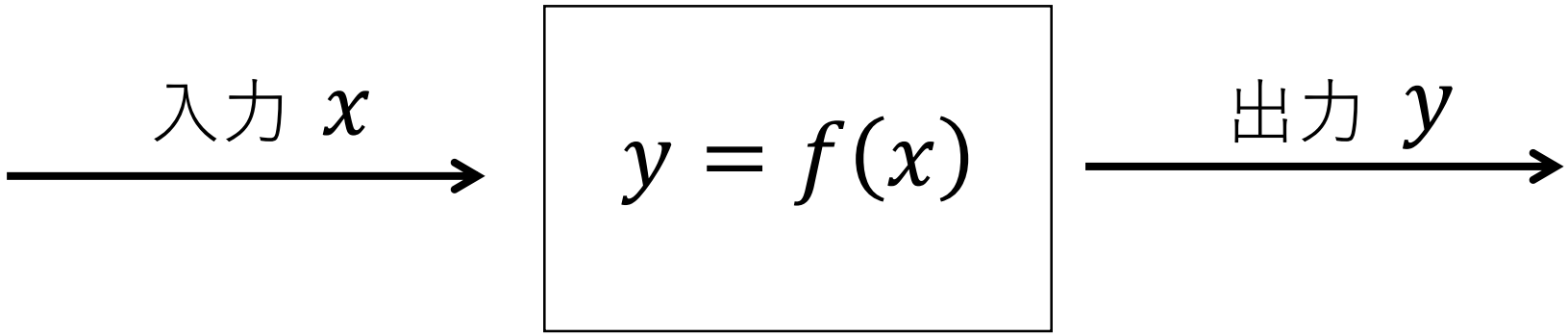
- ディープラーニングを用いた手書き文字(数字)認識の実例

ニューラルネットワークモデルの計算ルール

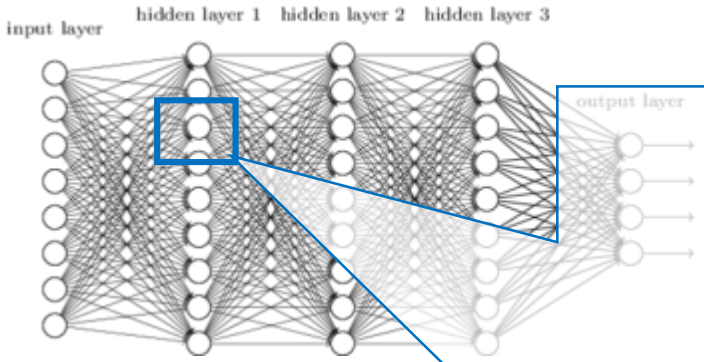


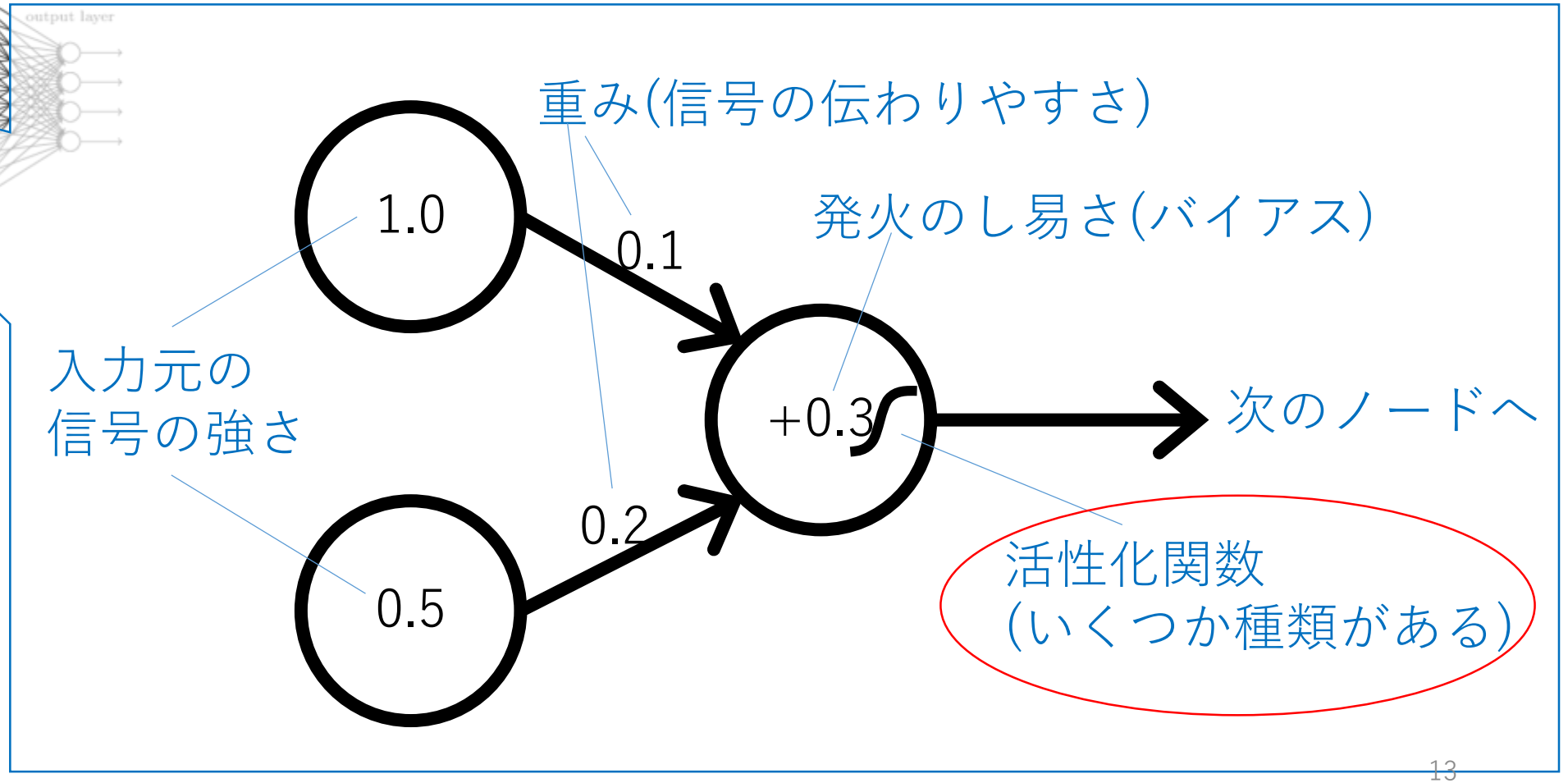


「関数」ってなに？



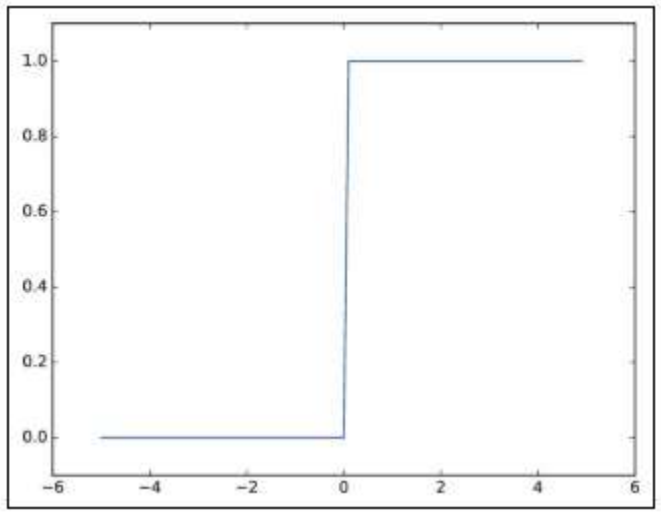
ニューラルネットワークモデルの計算ルール





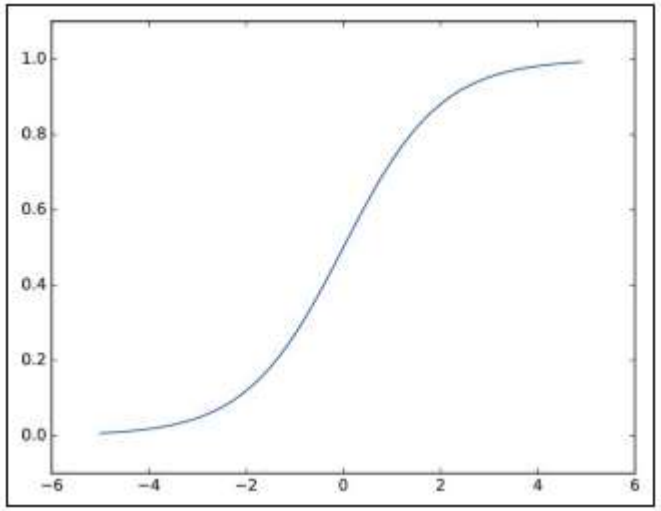
活性化関数

ステップ関数



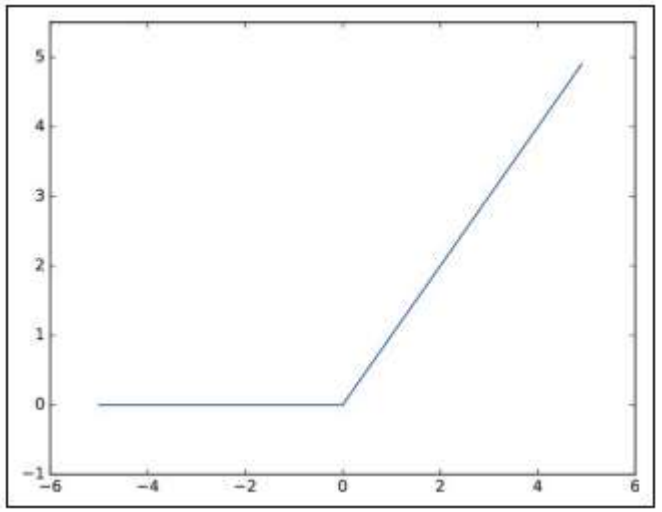
$$h(x) = \begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases}$$

シグモイド関数



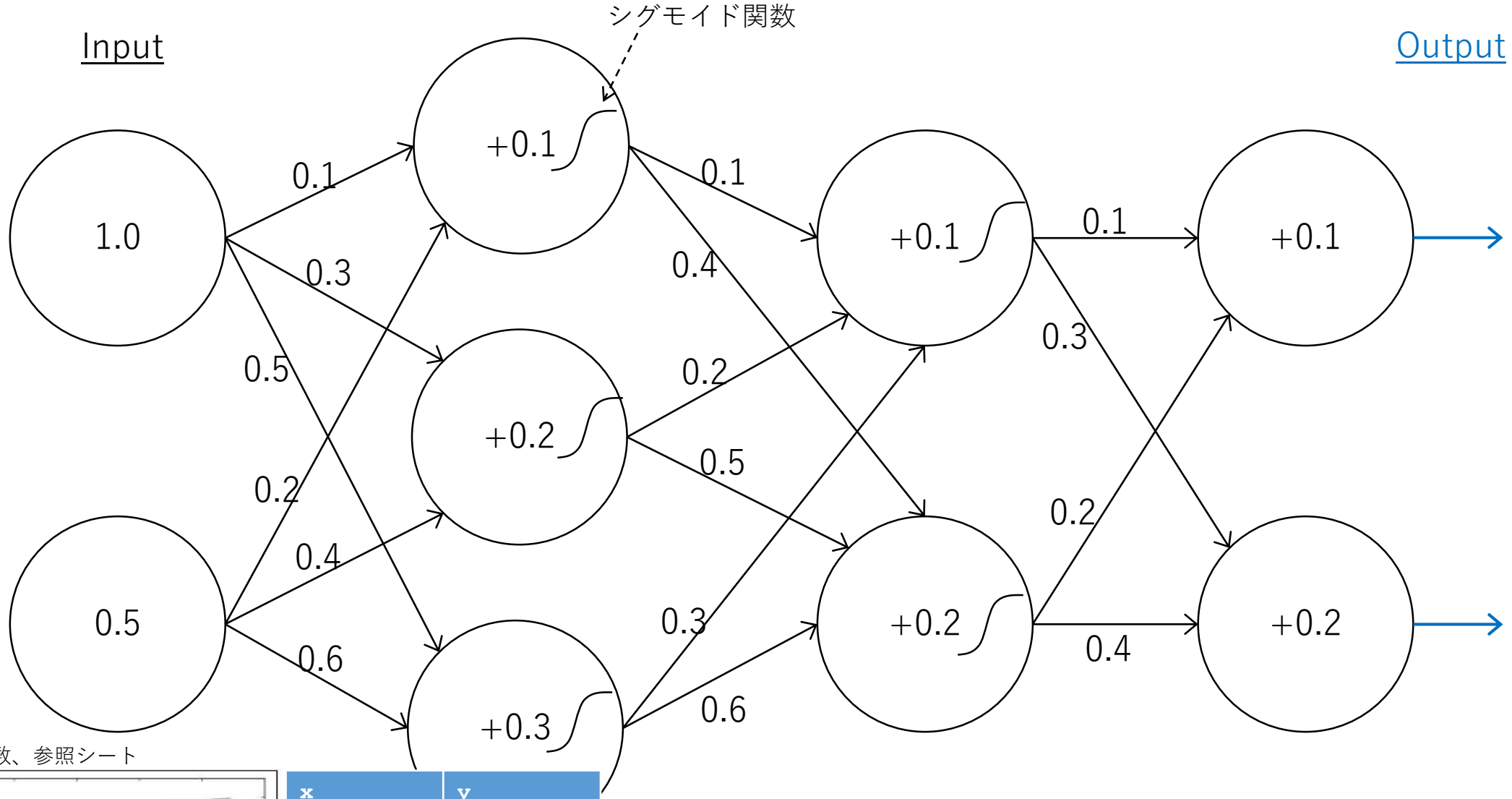
$$h(x) = \frac{1}{1 + \exp(-x)}$$

ReLU関数
(Rectified Linear Unit)

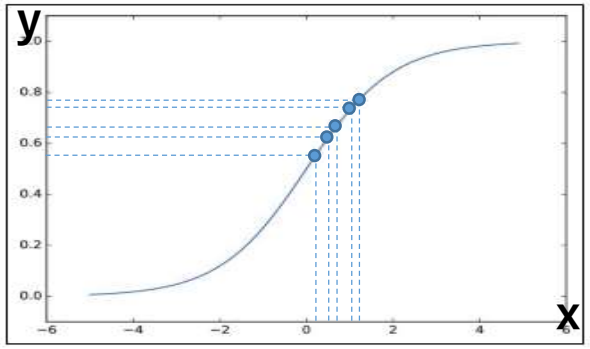


$$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$

手触り感持って理解するために、さほどディープじゃないニューラルネットワークモデルを使って手計算してみよう

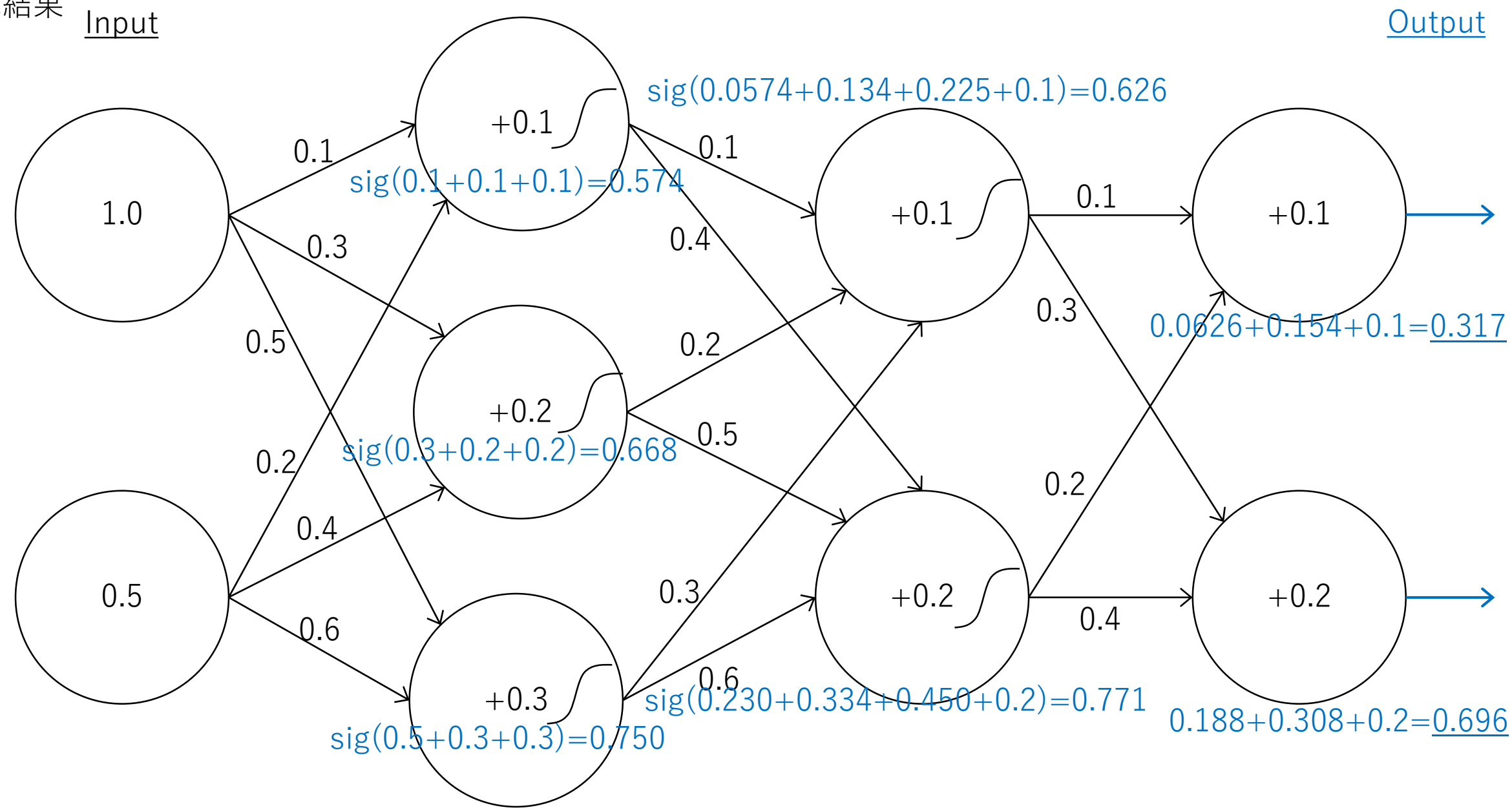


シグモイド関数、参照シート

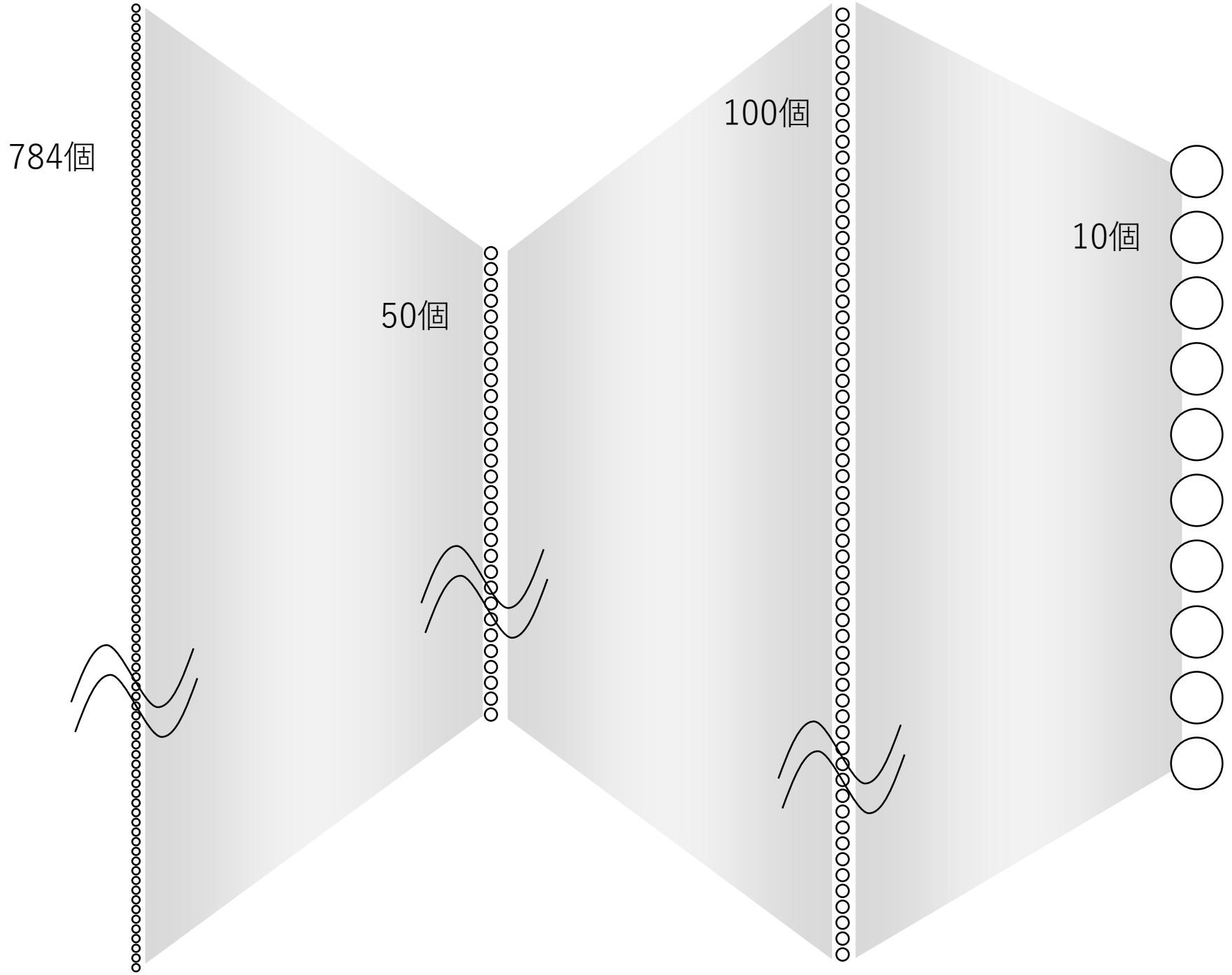


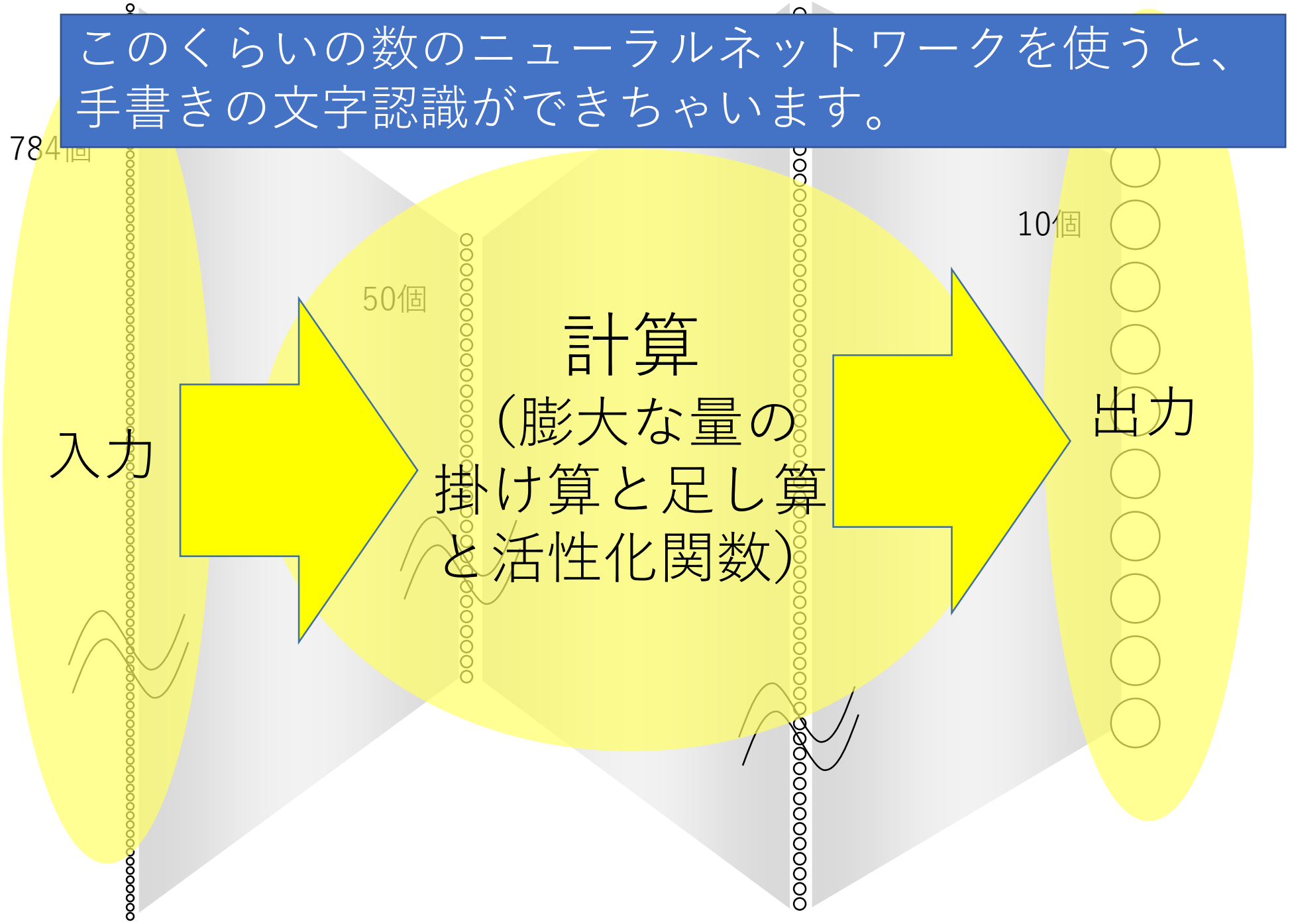
x	y
0.3	0.574
0.516	0.626
0.7	0.668
1.1	0.750
1.214	0.771

手計算結果



sig()：シグモイド関数


784個
50個
100個
10個


このくらいの数のニューラルネットワークを使うと、
手書きの文字認識ができちゃいます。
784個
50個
10個
入力
計算
(膨大な量の
掛け算と足し算
と活性化関数)
出力

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！

今日のアジェンダ

- 今日の立ち位置、言葉の整理

- ディープラーニングの
  - 計算のやり方
  - 計算の道具

- ディープラーニングを用いた手書き文字(数字)認識の実例

ふたつの道具を手に入れよう

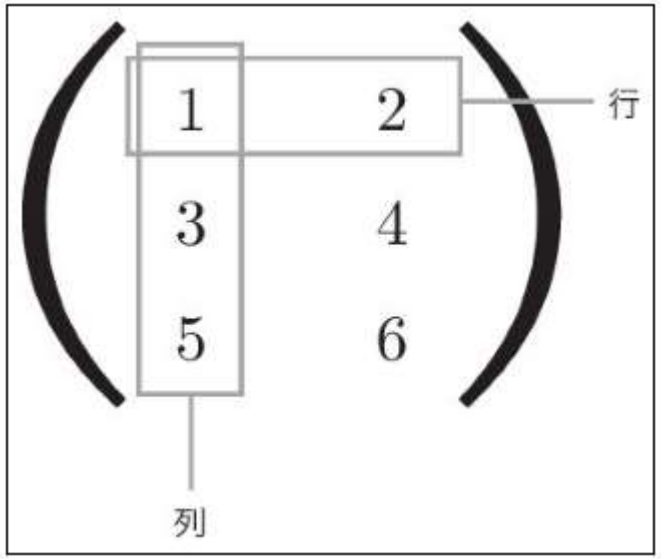
- 数学の道具 – 行列の計算
- プログラミング – Python

（なぜPythonかは後ほど補足）

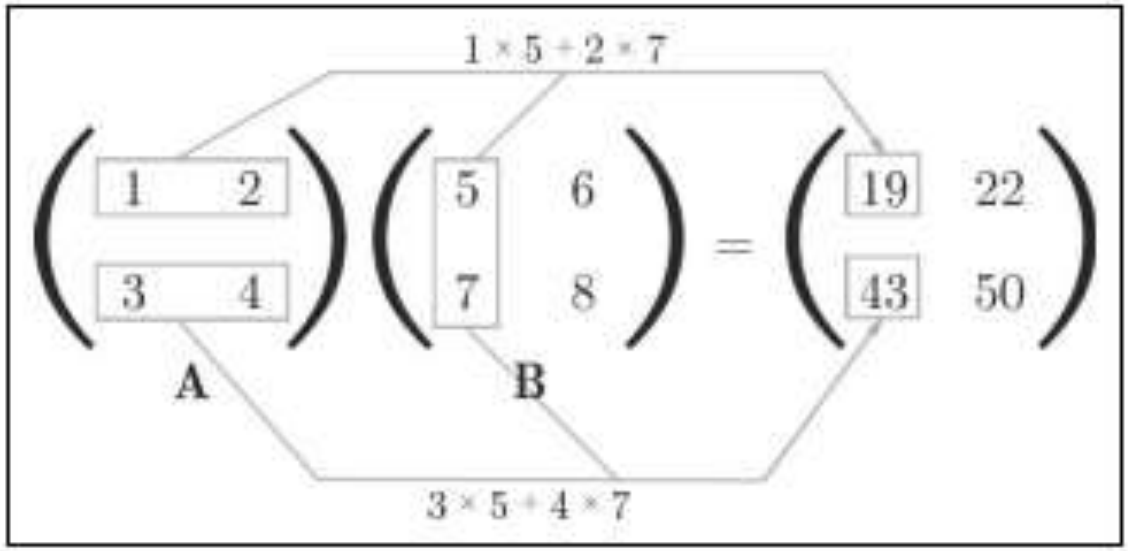
数学の便利な道具 – 行列、内積

なぜ便利かは後ほどわかる

3 x 2の行列の例



行列の内積の例



行列計算に慣れ親しむ

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

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


Break

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

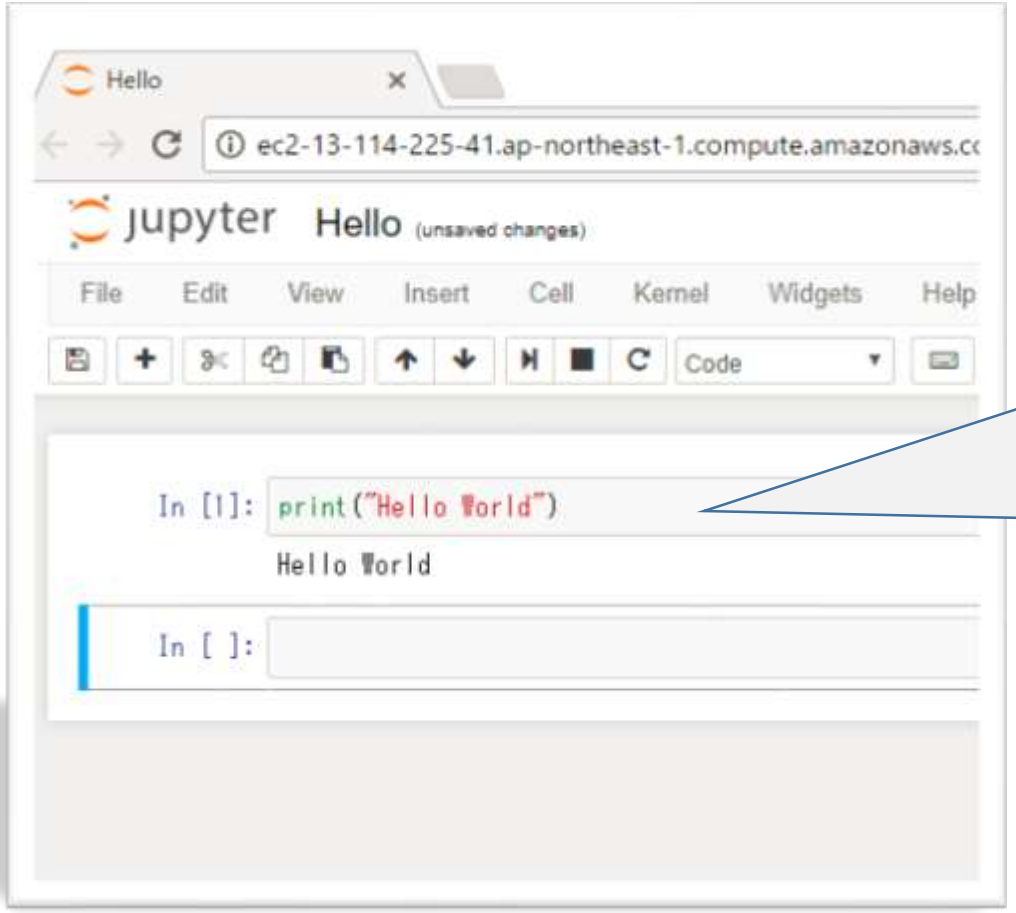
- ベクトル　＝　配列　＝　リスト

  数学っぽい　プログラム‥　特にPython

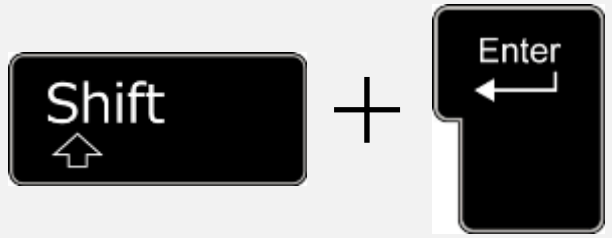
表記方法

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

いよいよPythonプログラミングです



Jupyterというブラウザベースの開発環境を使います。



で、セル内のコマンド(プログラム)を実行します。

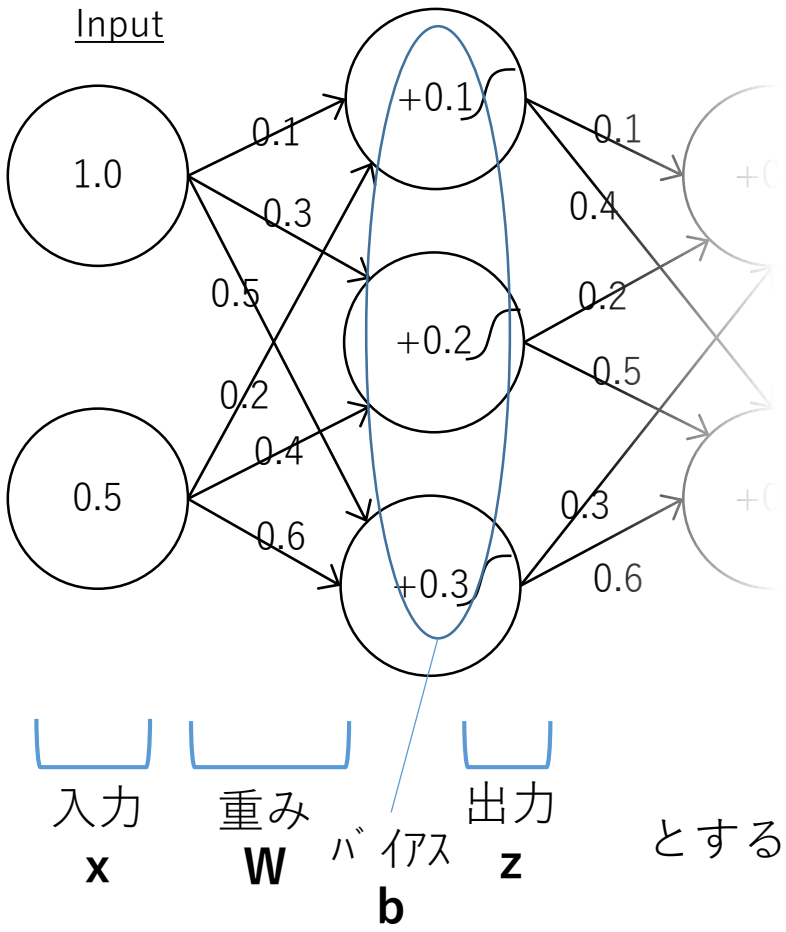
なぜ、行列（内積）なんていう道具を使うのか？

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



とすると、

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

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

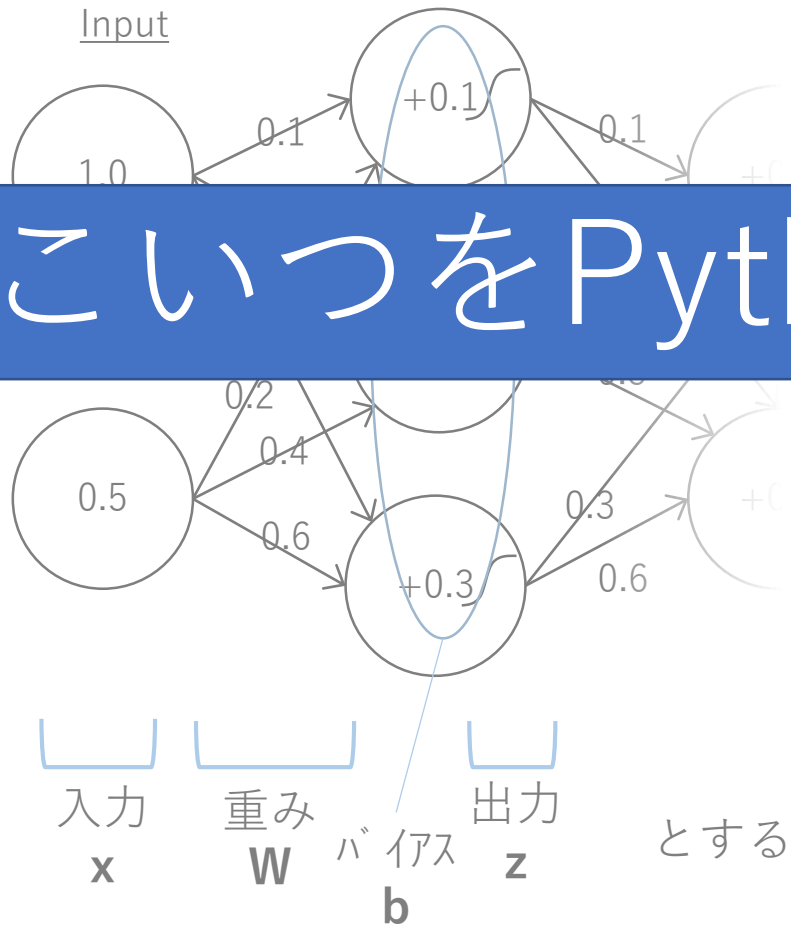
$$= \mathbf{sig}\left(\begin{pmatrix} 0.2 \\ 0.5 \\ 0.8 \end{pmatrix} + \begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix} \right)$$

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

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



とすると、

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

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

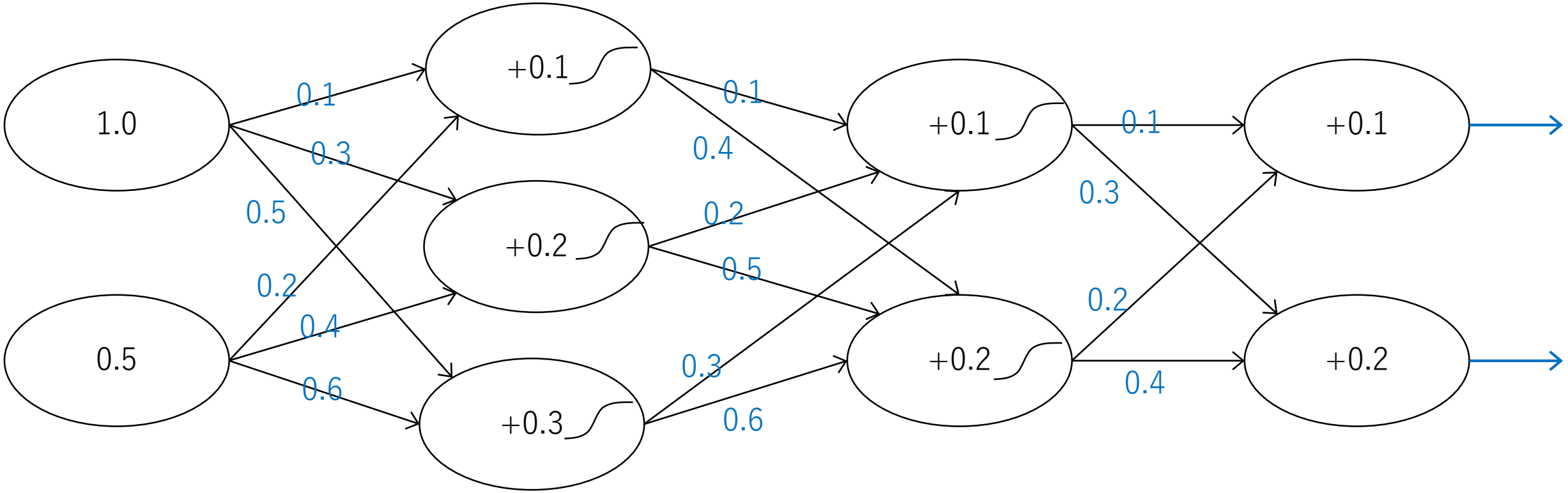
$$= \mathbf{sig}\left(\begin{pmatrix}0.2\\ 0.5\\ 0.8\end{pmatrix} + \begin{pmatrix}0.1\\ 0.2\\ 0.3\end{pmatrix}\right)$$

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

こいつをPythonで実装してみよう！

先ほどのニューラルネットの例

Jupyterを使う



x

$$\begin{pmatrix} 1.0 \\ 0.5 \end{pmatrix}$$

W1

$$\begin{pmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \\ 0.5 & 0.6 \end{pmatrix}$$

b1

$$\begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix}$$

z1

$$\begin{pmatrix} ? \\ ? \\ ? \end{pmatrix}$$

W2

$$\begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \end{pmatrix}$$

b2

$$\begin{pmatrix} 0.1 \\ 0.2 \end{pmatrix}$$

z2

$$\begin{pmatrix} ? \\ ? \end{pmatrix}$$

W3

$$\begin{pmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{pmatrix}$$

b3

$$\begin{pmatrix} 0.1 \\ 0.2 \end{pmatrix}$$

z3

$$\begin{pmatrix} ? \\ ? \end{pmatrix}$$

z1=sig(W1・x+b1)

z2=sig(W2・z1+b2)

z3=W3・z2+b3

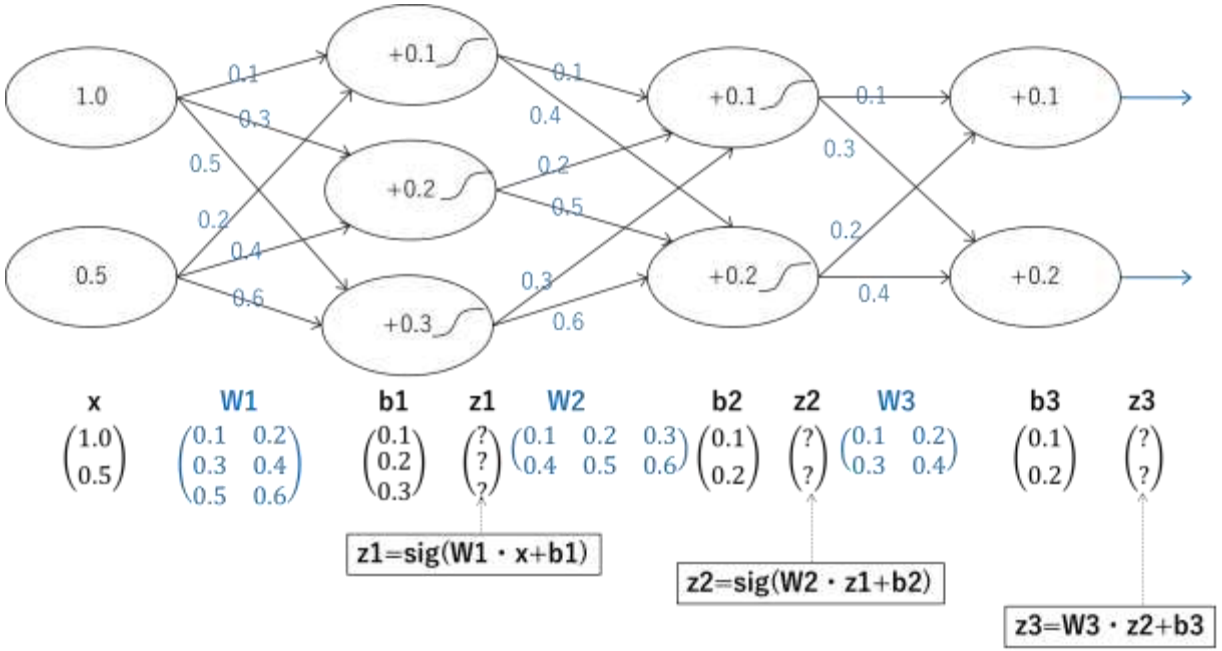
sig() …シグモイド関数

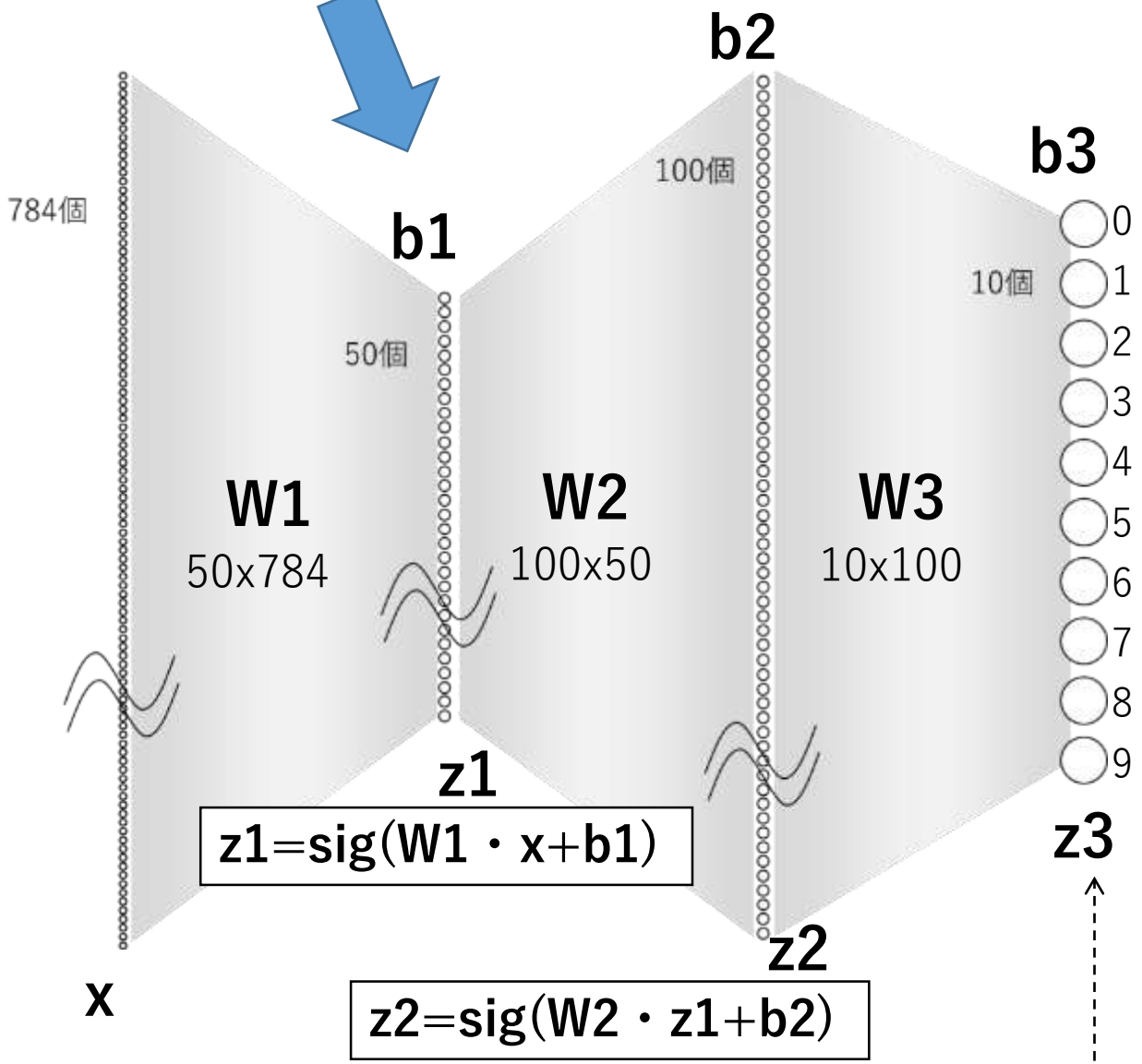
```
import numpy as np

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！

先ほどやったやつ

だったらこいつもいけるんじゃね？





層やノードの数が増えても、行列を使うとシンプルに表現できる!
そしてPythonプログラムで記述できる。

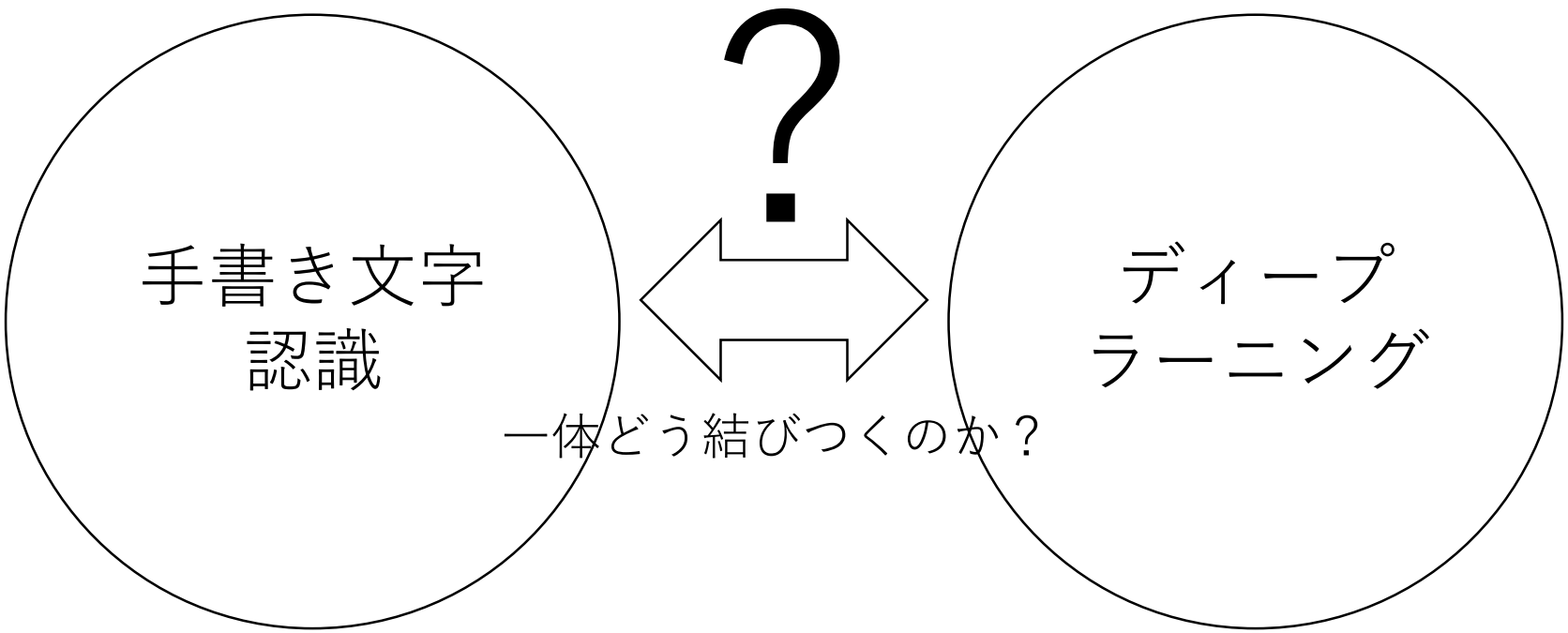
今日のアジェンダ

- 今日の立ち位置、言葉の整理

- ディープラーニングの
  - 計算のやり方
  - 計算の道具

- ディープラーニングを用いた手書き文字(数字)認識の実例

手書き文字(数字)認識をさせてみる



人間が文字認識する、をホワイトボードでやってみる

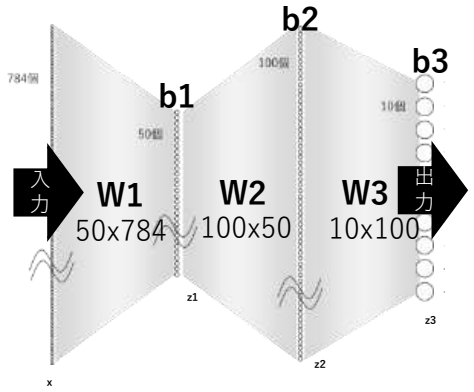
ディープラーニングの全体の流れ

データの準備

MNIST手書き文字データ
http://yann.lecun.com/exdb/mnist/

- 6万文字分の学習用データ
- 1万文字分の検証用データ

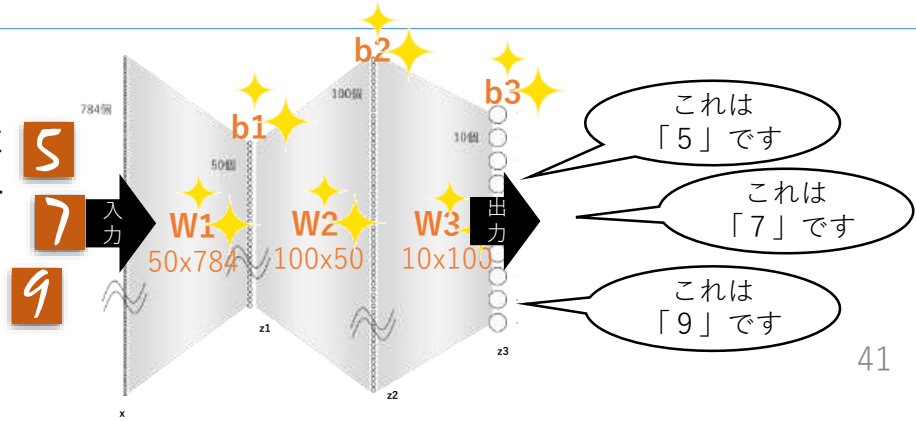
学習



6万文字分の学習用データを使って、入力した手書き文字に対応した出力が得られるようにパラメータW1, W2, W3, b1, b2, b3 を調整

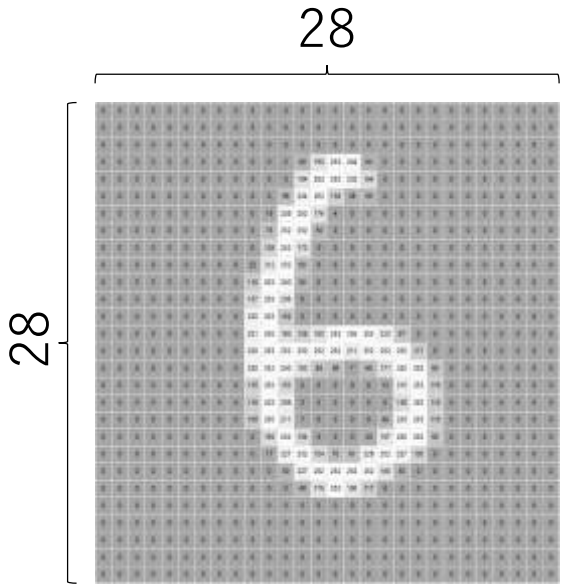
検証／適用

学習済(うまくパラメータが調整された状態)のニューラルネットに検証用データを入力し、うまく認識されることを検証する



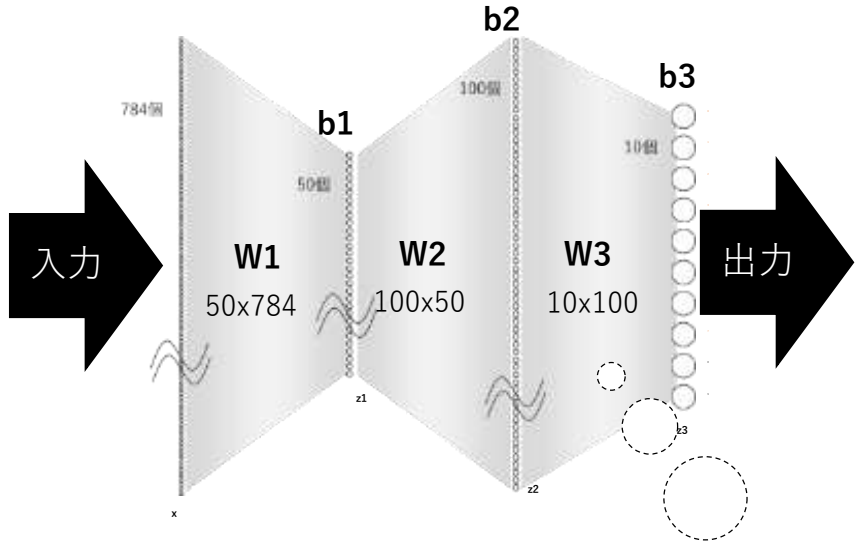
ディープラーニングにおける学習とは

手書き文字を数値化し、



28x28=784個の格子(ピクセル)ごとに0~1の255段階の値で明るさを示すことで手書き文字を表現

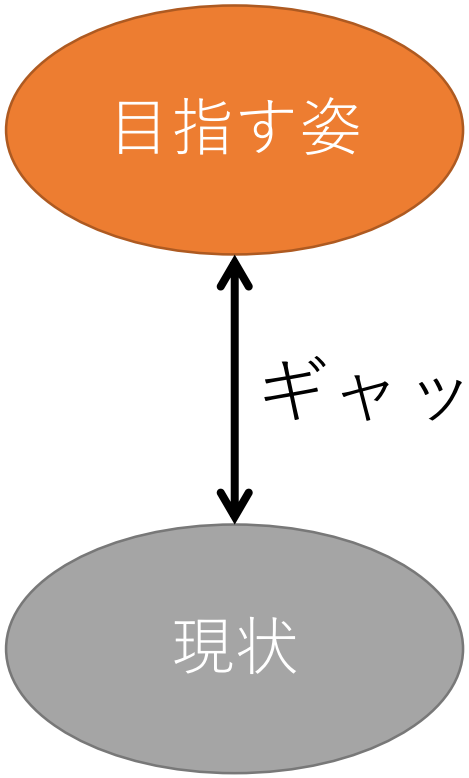
ニューラルネットに食わせ‥



4.9
-2.2
-1.3
-1.1
2.7
4.1
7.6
-3.2
5.3
3.1

パラメータ(W1, W2, W3, b1, b2, b3 –全部で45,350個の数字)を少しずつ変えながら、入力に対応した箇所が大きな数値を示すような**絶妙な組み合わせを探す**プロセス

具体的にどうアプローチするか—指標の定義



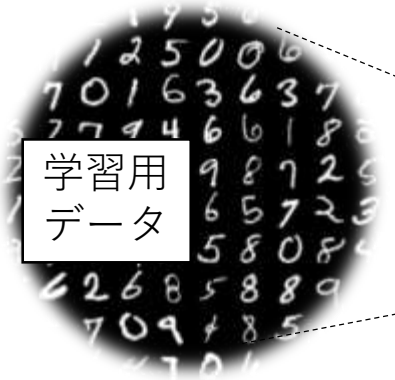
完璧に数字を判別できる状態

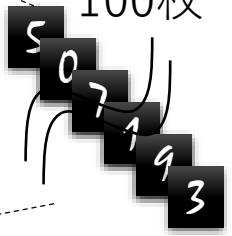
ギャップを示す指標(損失関数)を定義する→

この損失関数が最小となるパラメータW1, W2, W3, b1, b2, b3 の組み合わせを探す

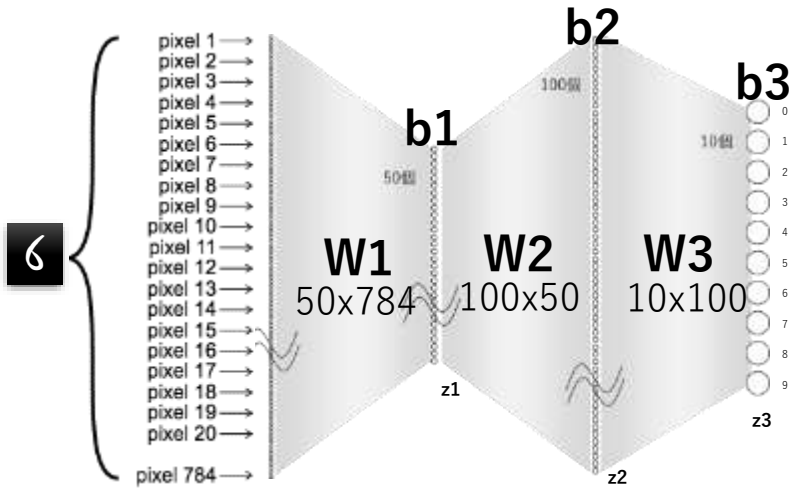
学習の途中段階

損失関数～当たってなさ具合の指標

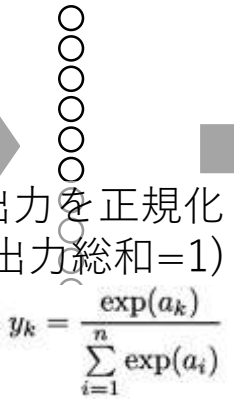




MNIST計6万枚のデータ群から、
ランダムに100枚を選び出す。



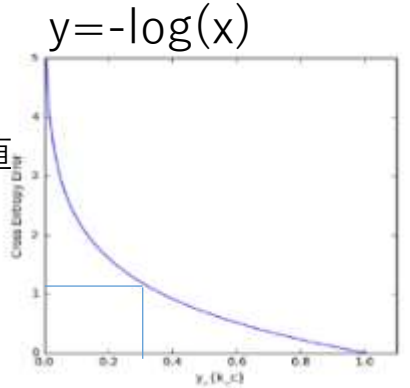
一枚ずつニューラルネットに放り込んで計算



出力を正規化
(出力総和=1)

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

正解箇所の値
のｴﾝﾄﾛﾋﾟｰ
-logを計算



100枚分計算して平均を求める。
これが<u>損失関数の値</u>となる。

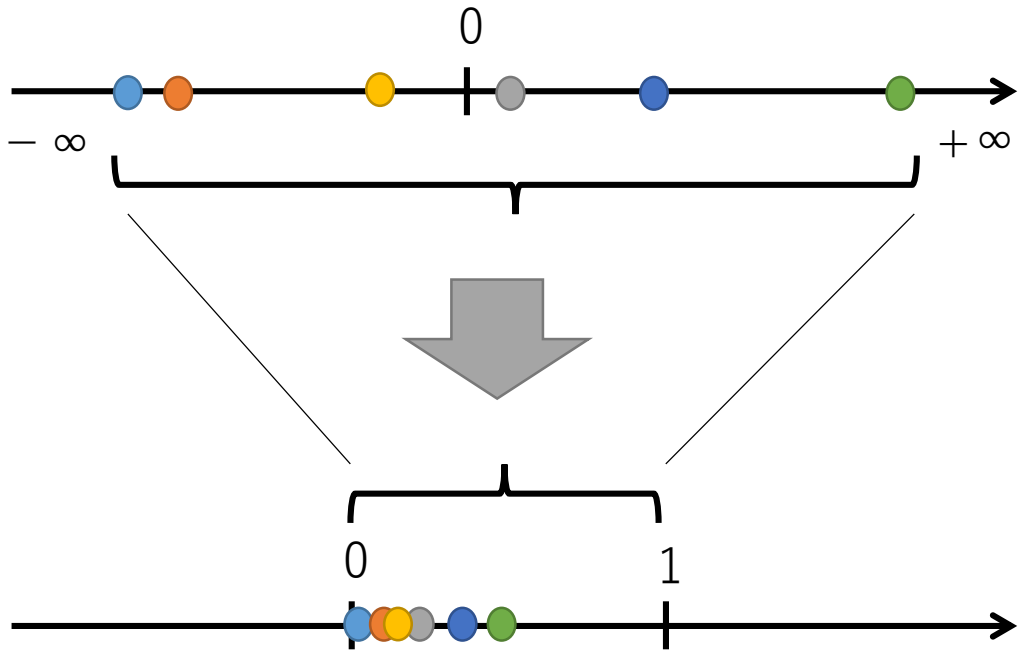
パラメータW1, W2, W3, b1, b2, b3
のある組み合わせ(学習の途中段階)
における当たってなさ具合

（補足）指数関数を用いた正規化

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

正規化の様子を数直線で表現すると、
大小さまざまな数字について、それぞれの
位置関係は保ったままで、

- 0から1のあいだにギュッと押し込む
- 且つ、値の総和が1になる



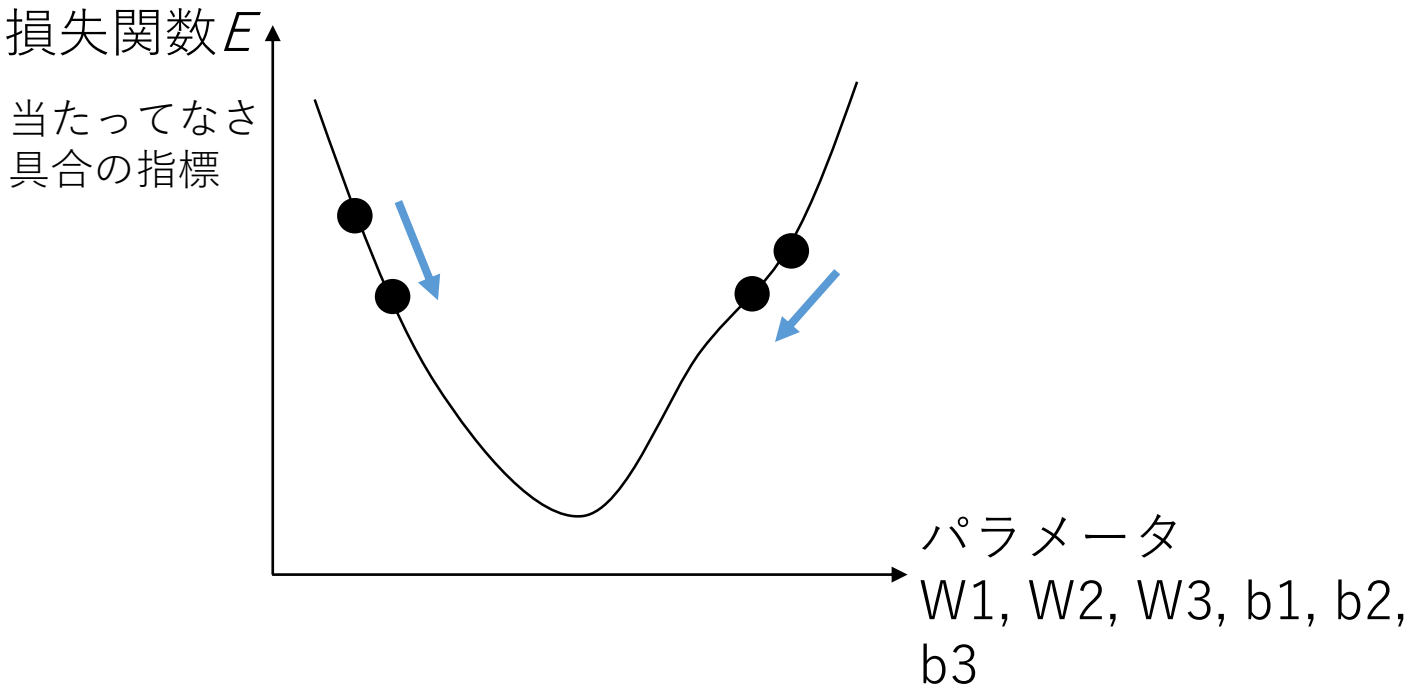
Excelシートで実際に試してみた例

適当な数字
=EXP(左セル)
左セルを%表示

−1.2	0.301194	0.037784	4%
−0.3	0.740818	0.092934	9%
1.3	3.669297	0.460304	46%
0.1	1.105171	0.138641	14%
−1.1	0.332871	0.041758	4%
0.6	1.822119	0.22858	23%
	7.97147		100%

=SUM(列の合計)
=(左セル)÷SUMの値
列のSUM
確かに足したら1

損失関数が最小となるパラメータを探す

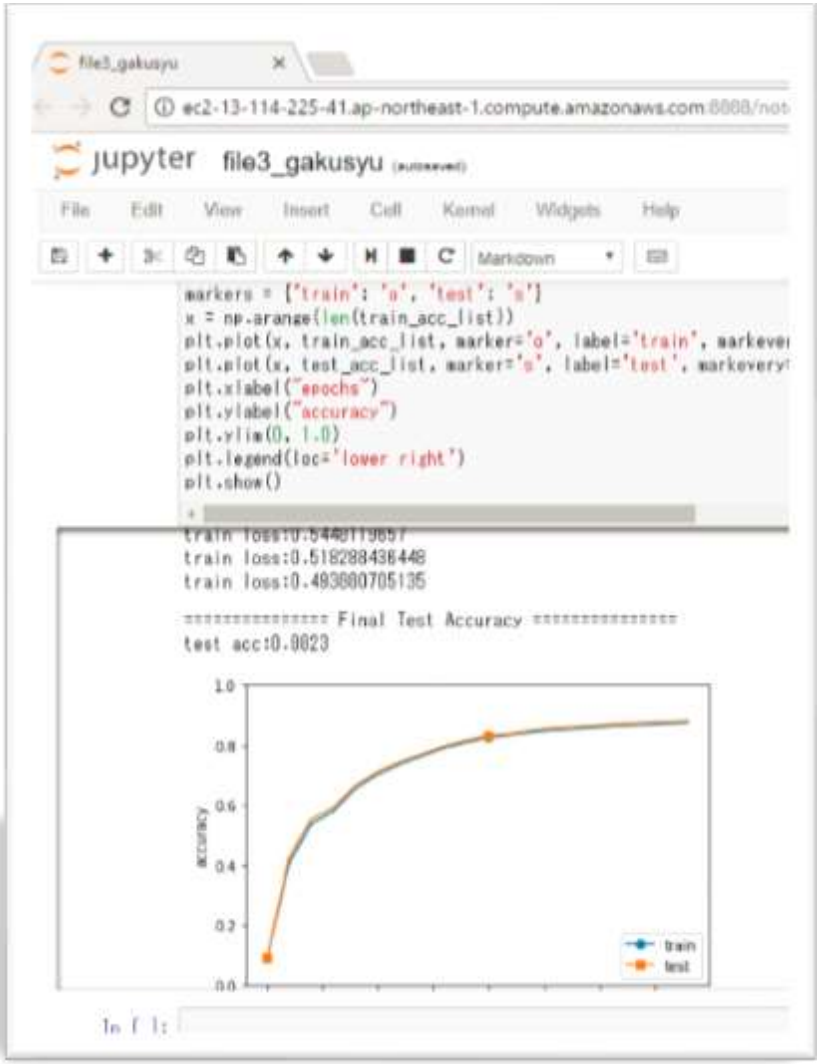


$$W \underset{\text{代入}}{\leftarrow} W - \alpha \frac{\partial E}{\partial W}$$

$$b \underset{\text{代入}}{\leftarrow} b - \alpha \frac{\partial E}{\partial b}$$

足元の坂の傾きを調べて、
その傾きの大きさに従って一歩進む。
それを繰り返して、Eの一番低いところに
たどり着く

実際の学習の過程を見てみる



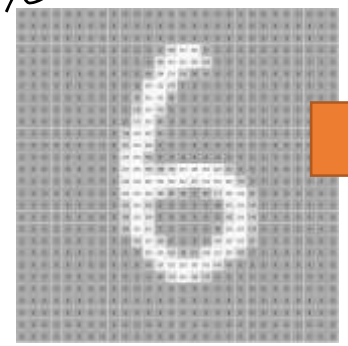
ディープラーニングの学習 ~ 奥深い世界

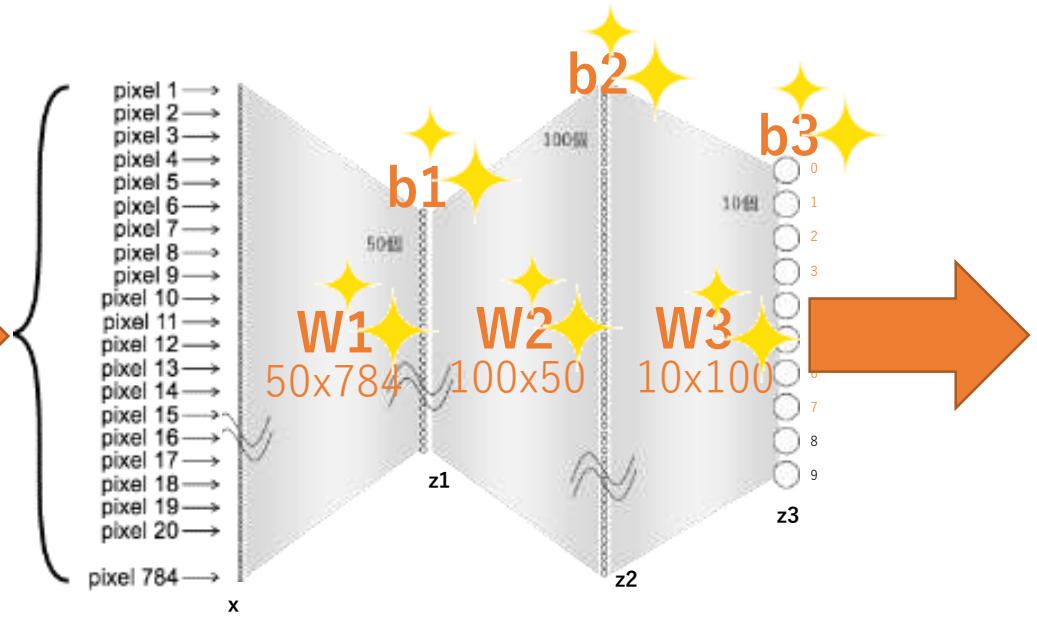
- 学習をいかに効率よく行うかが、実際にディープラーニングを使う上で大きな課題
- 学習(コンピューターの数値計算)の手法はそれ自体が奥深い研究テーマであり、誤差逆伝搬法(バックプロパゲーション)、SGD、Momentum、AdaGrad、Adam、…など、専門用語がバンバン出てくる領域。今日はそのあたりの深入りはやめときます

学習済のパラメータを使って、文字認識が正しく行われていることを確かめる

検証用のデータ

例えば「6」

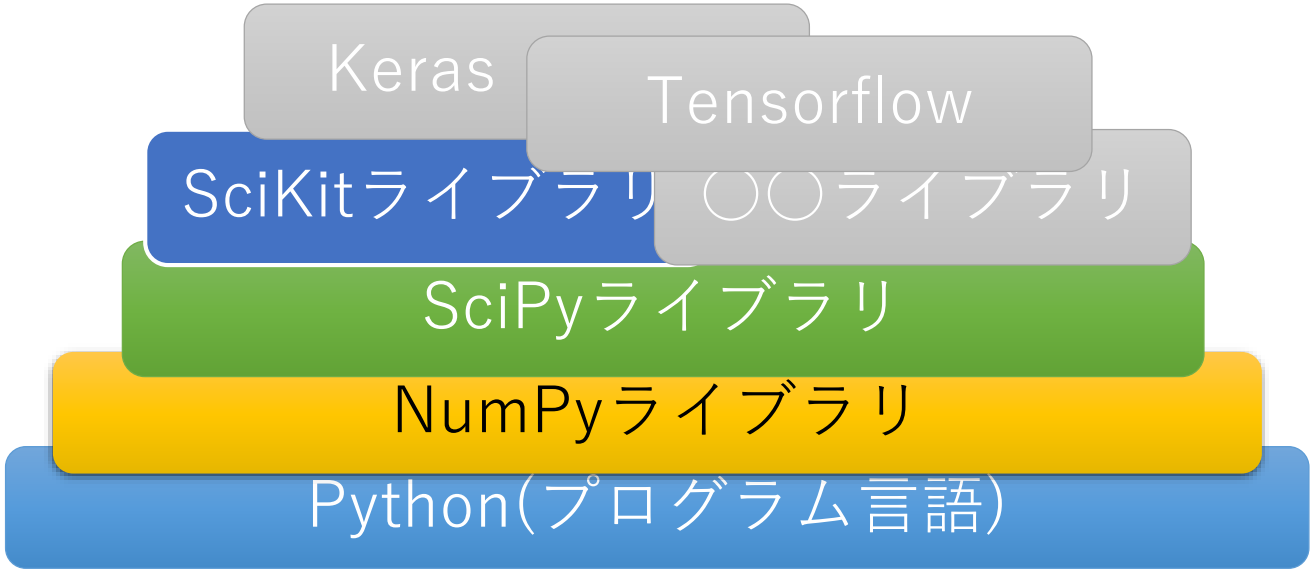




「6」と認識できるか？
実際にPythonで動かして試してみよう！

ということで、一通りのおさらい

（ちょいと脱線）
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

パラメータとハイパーパラメータ

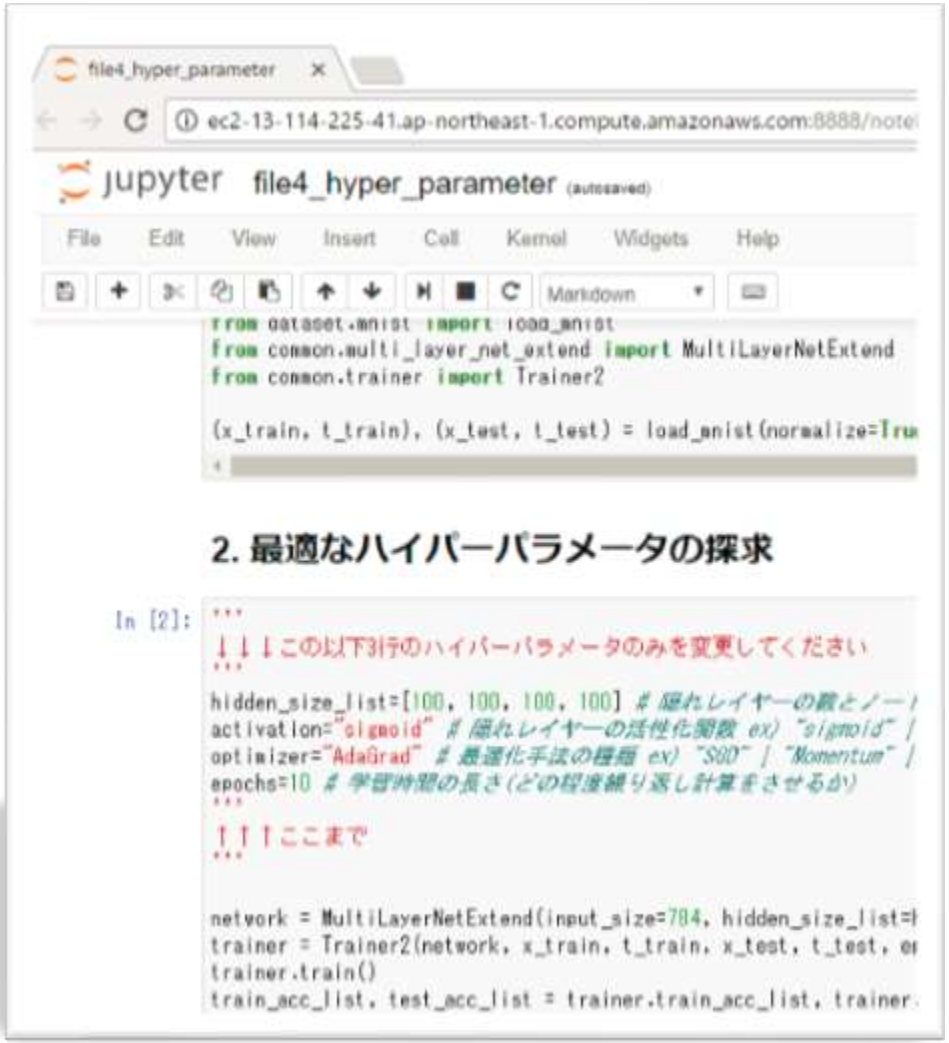
パラメータ
W1, W2, W3, b1, b2, b3 ⟵ 計算で求める
（全部で45,350個の数字）

ハイパーパラメータ
- 何層にするか
- 各層のノードの数
- 活性化関数の種類
- …

⟵ 人による設計
何度も計算させながら試行錯誤を繰り返す

より高いコンピュータの計算能力が求められる

ハイパーパラメータ探求を体感



パラメータの更新手法の数々

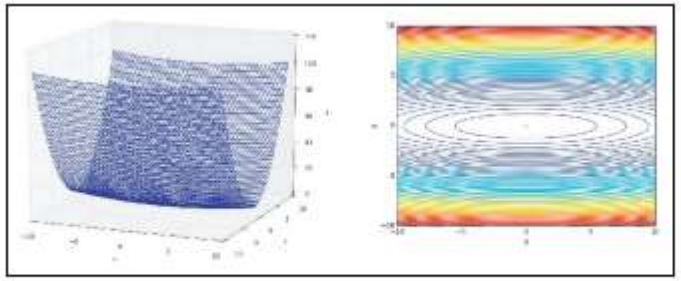


図6-1　$f(x,y)=\frac{1}{20}x^2+y^2$のグラフ（左図）とその等高線（右図）

SDG

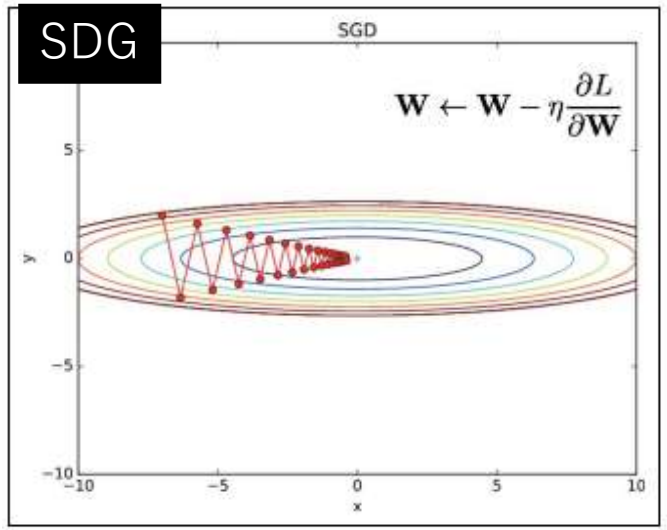


図6-3　SGDによる最適化の更新経路：最小値の(0, 0)へジグザグに動くため非効率

Momentum

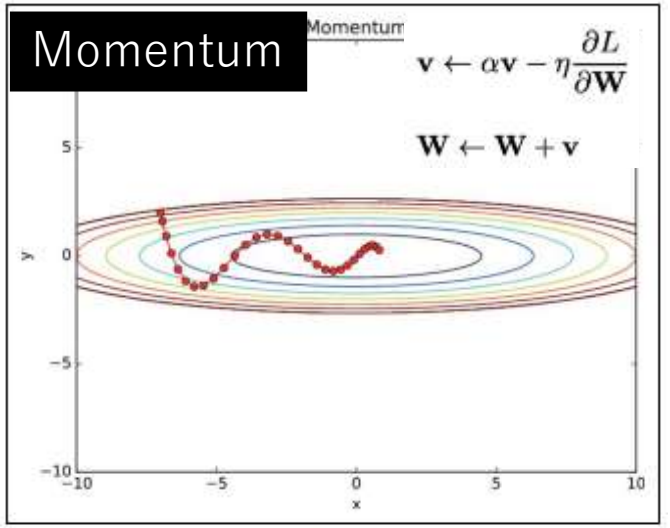


図6-5　Momentumによる最適化の更新経路

AdaGrad

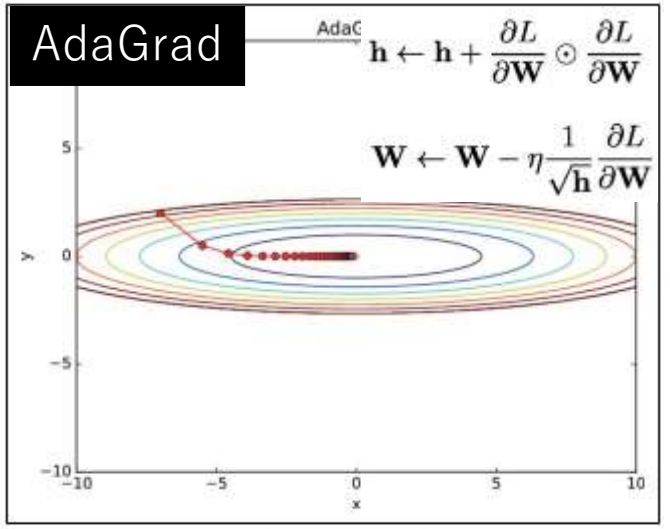


図6-6　AdaGradによる最適化の更新経路

Adam

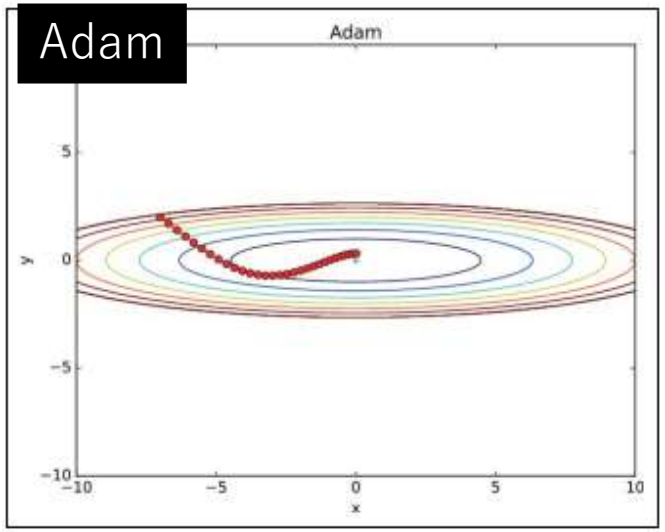
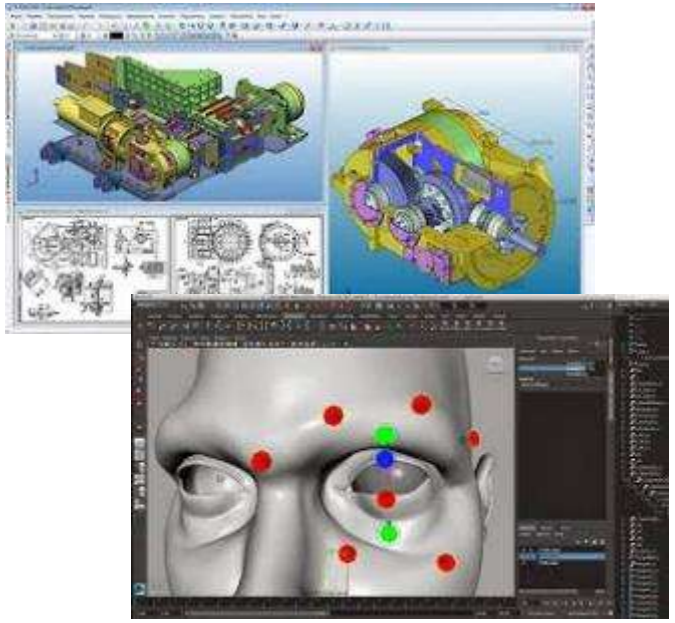


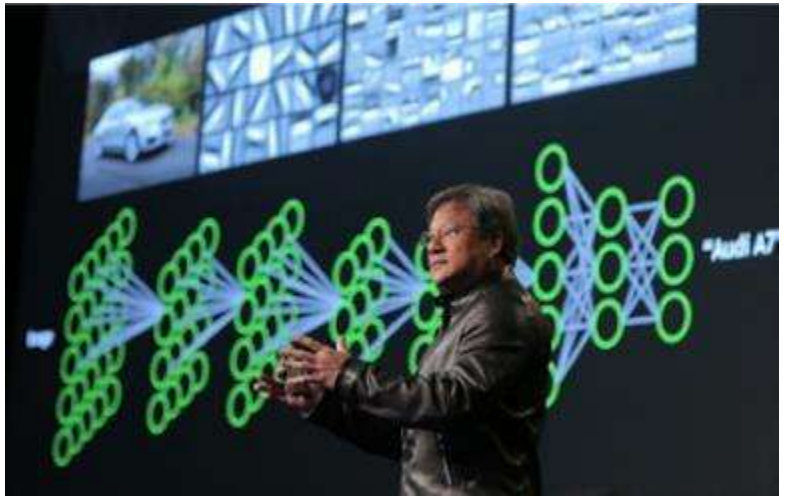
図6-7　Adamによる最適化の更新経路

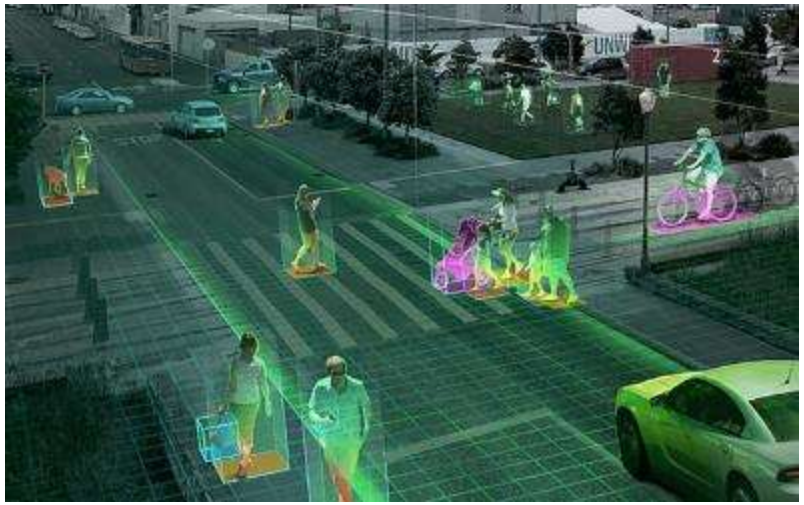
“謎のAI半導体メーカー”☺

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。

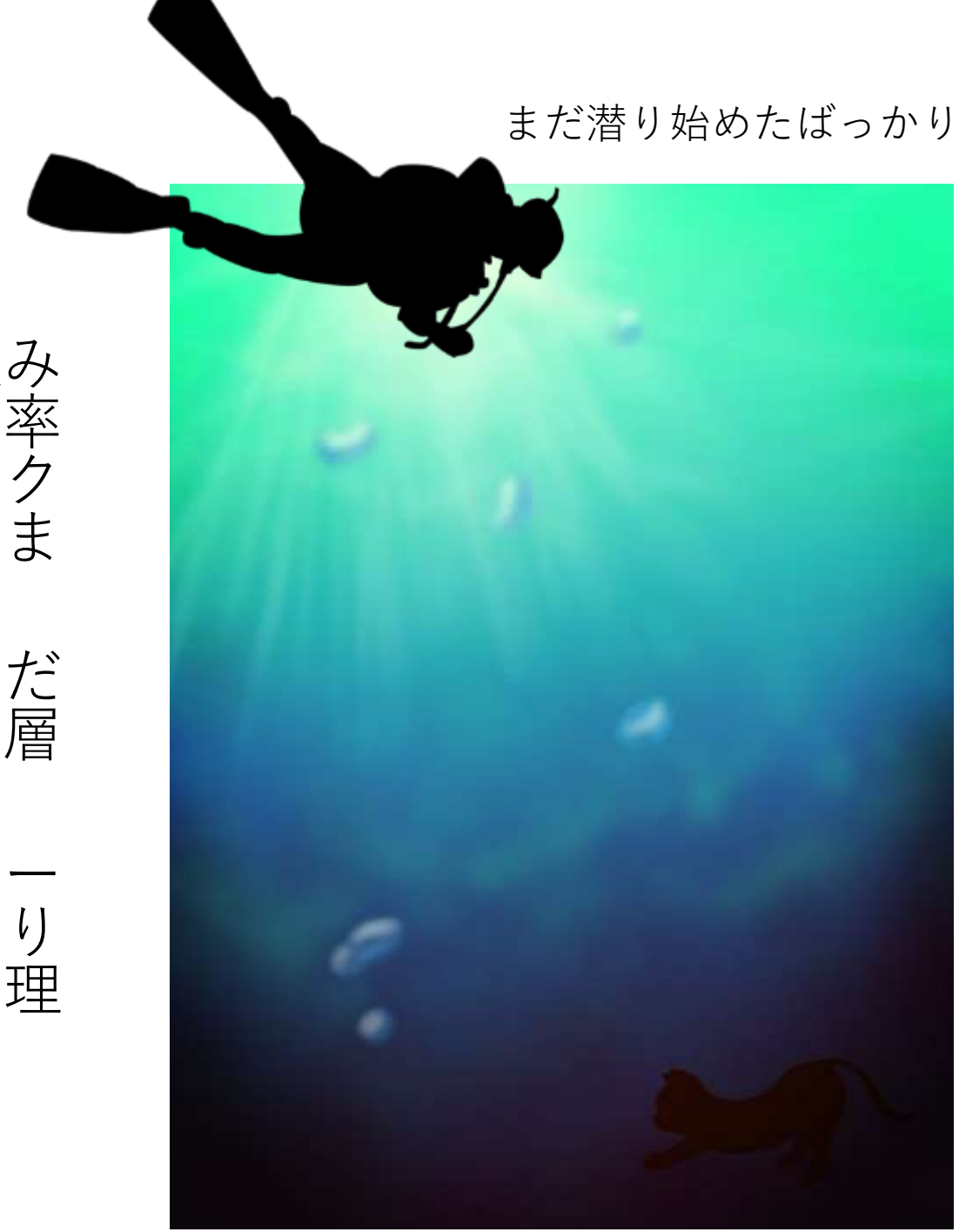






まだまだ先は深いけど

- 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



まだ潜り始めたばっかり

手書き文字認識

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