AI基礎講座
プログラミング体験
(初心者向け)

2019年11月22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。祖父母・両親・姉、親族 ほぼ皆 美容師
鎌倉/東京育ち、名古屋17年のあと、この4月から東京復帰

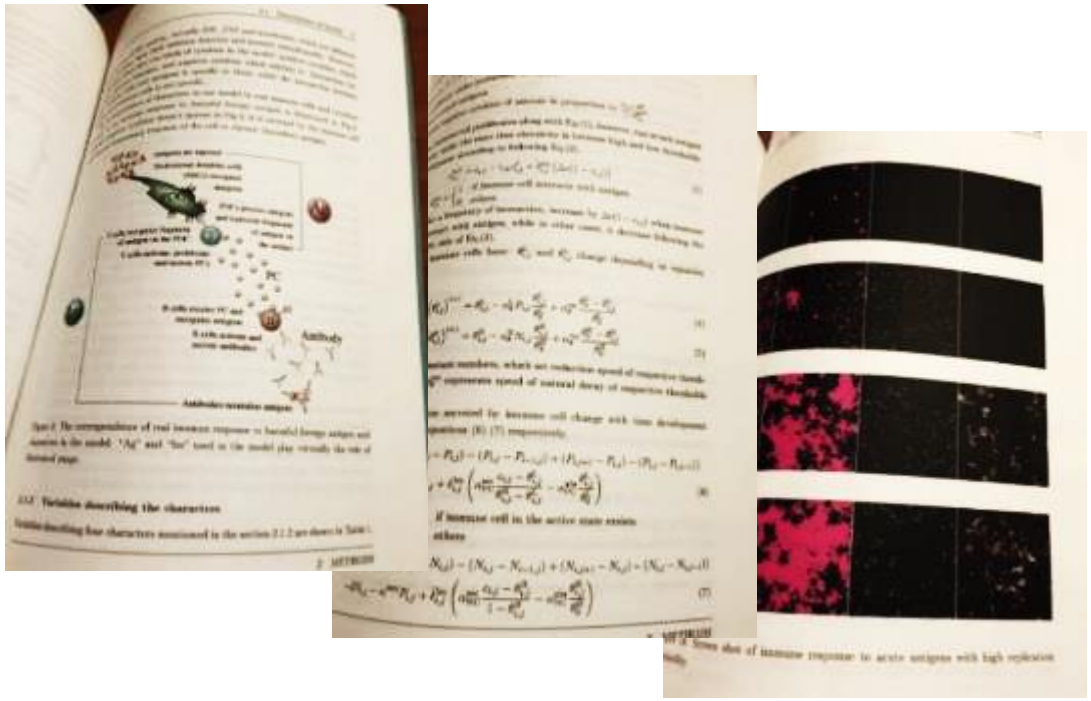
1998年～日本アイ・ビー・エム
2012年～ネットアップ 名古屋支店長
2018年～　〃　グローバル営業本部長

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

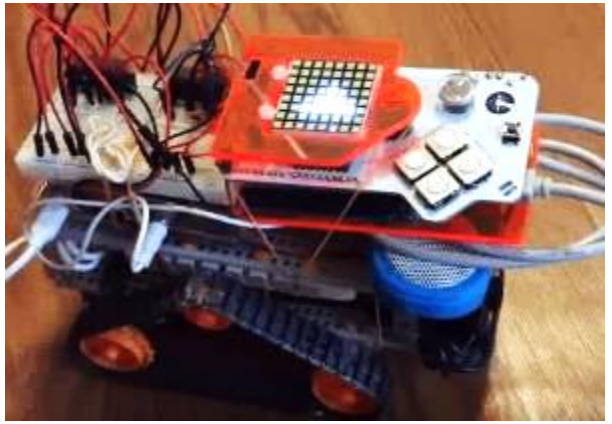
私とプログラミング

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



しばらく前、
オタクっぽい趣味の再燃



今日やること（やれるとこまで）

- プログラミング言語「Python(パイソン)」で簡単なプログラミングを打ち込んで動かしてみる

- ディープラーニングに関連するPythonのプログラムを打ち込んで動かしてみる

- 「手書き文字認識」を題材に、ディープラーニングの学習と推論の様子を実際に動かして見てみる

とにもかくにも

https://jupyter.org/try

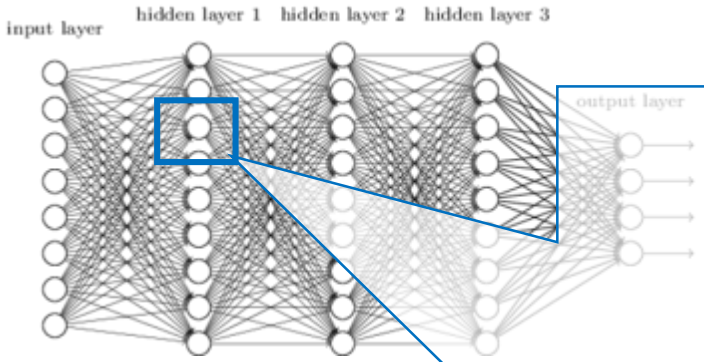
写経する

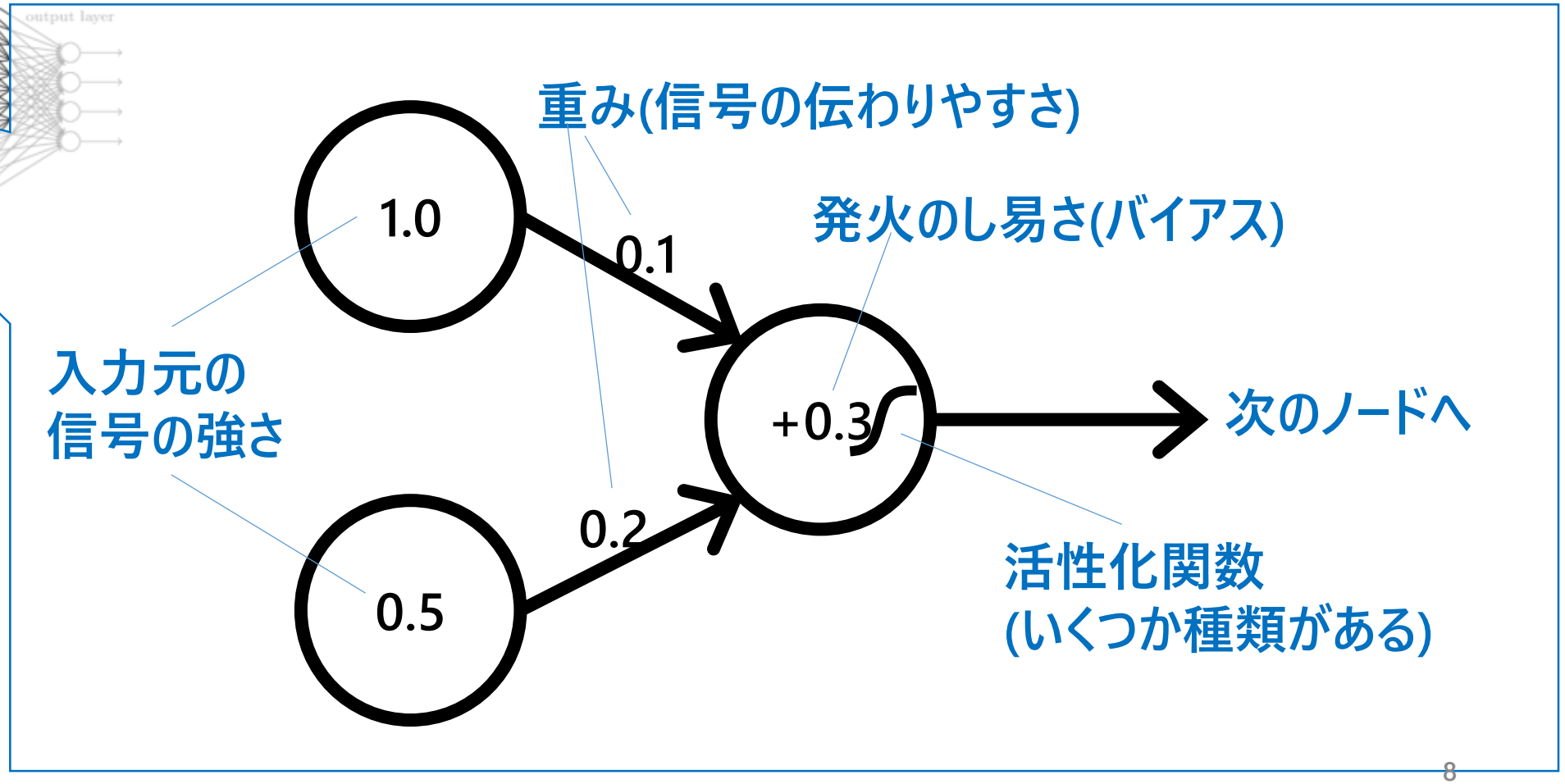
https://github.com/yoshihiroo/programming-workshop/tree/master/Python-basic/src

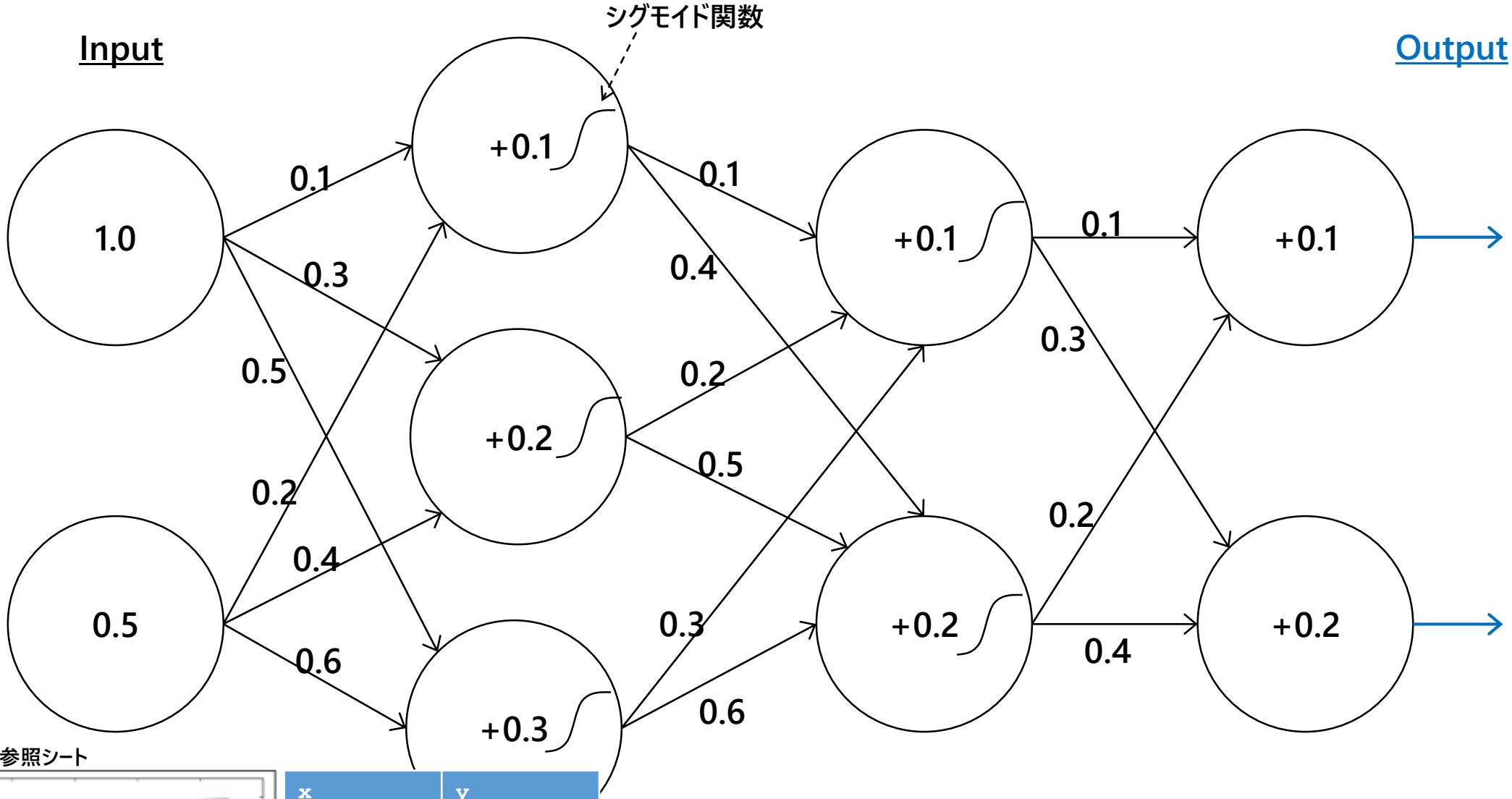
今日やること（やれるとこまで）

- プログラミング言語「Python(パイソン)」で簡単なプログラミングを打ち込んで動かしてみる
- ディープラーニングに関連するPythonのプログラムを打ち込んで動かしてみる
- 「手書き文字認識」を題材に、ディープラーニングの学習と推論の様子を実際に動かして見てみる

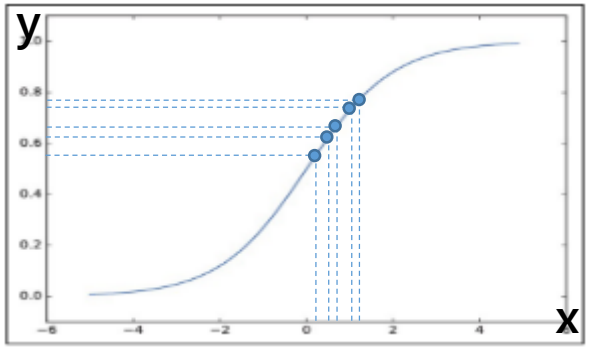
ニューラルネットワークモデルの計算ルール





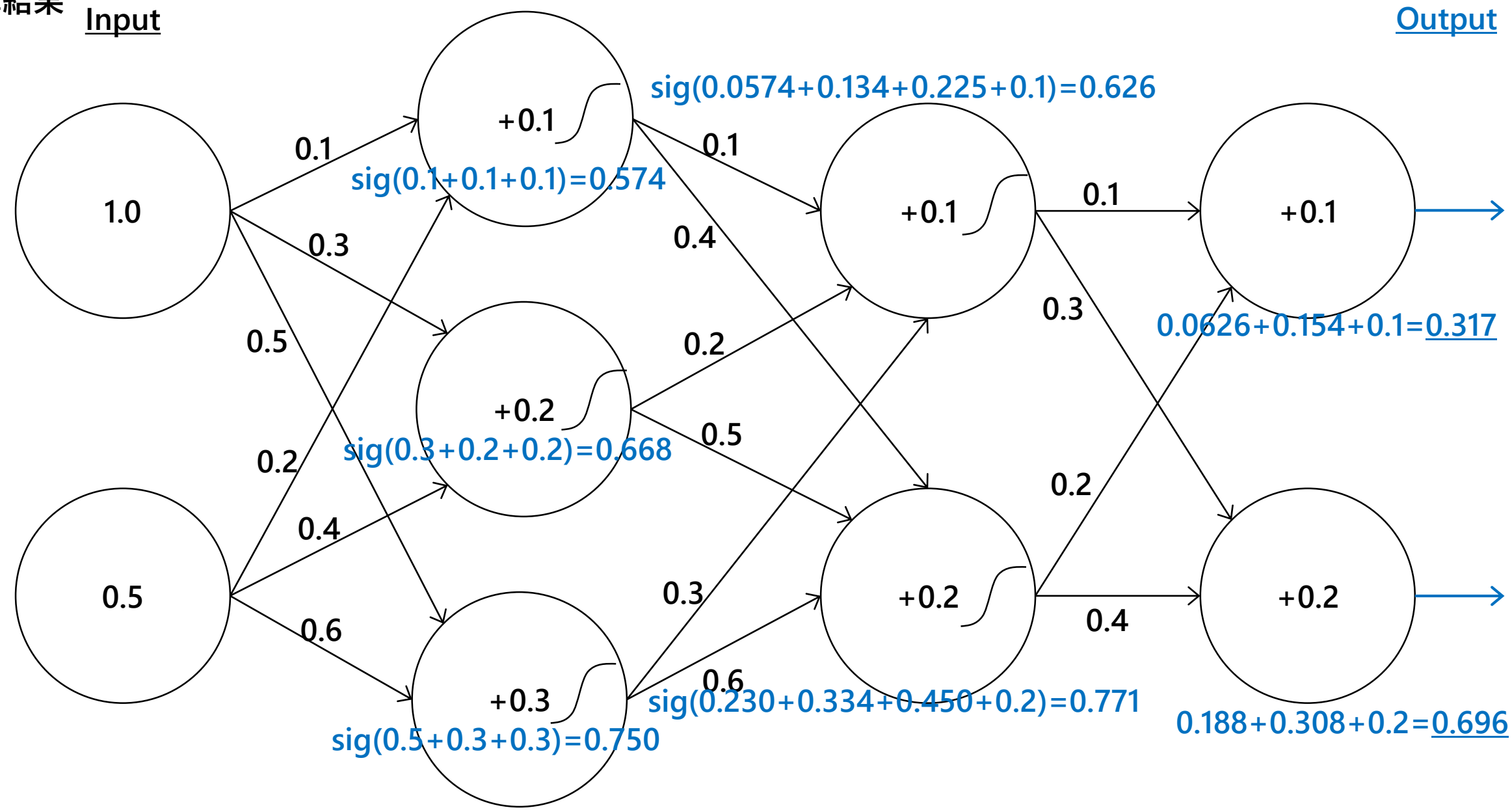


シグモイド関数、参照シート



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

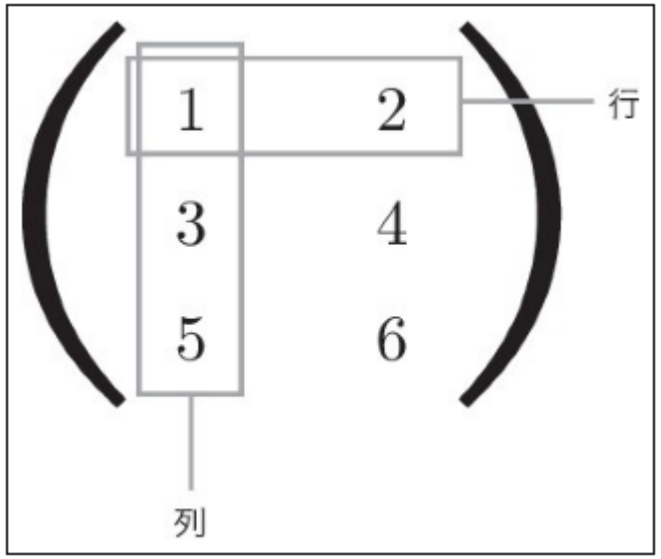


sig()：シグモイド関数

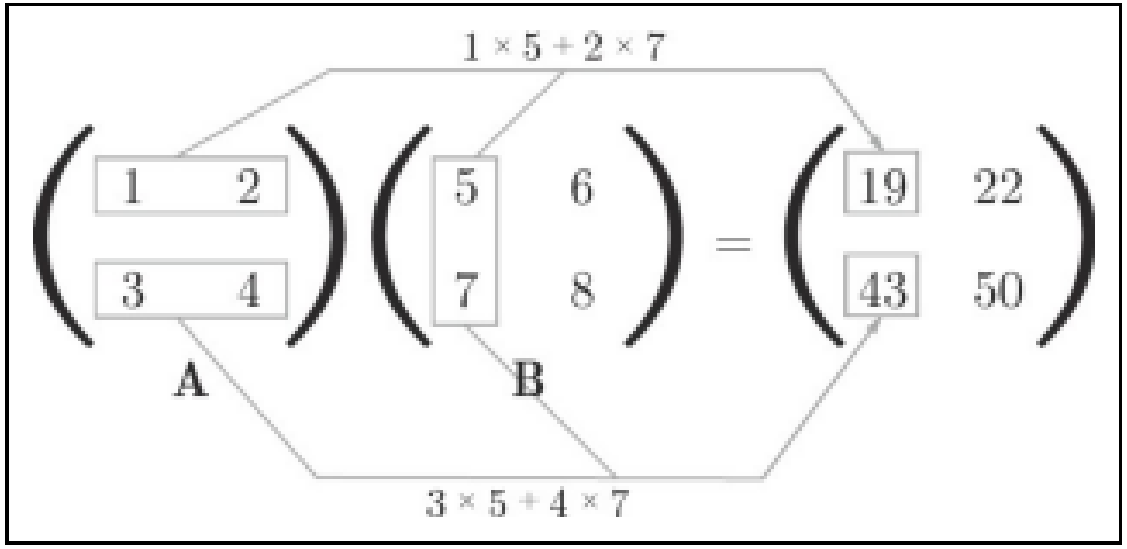
数学の便利な道具 – 行列、内積

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



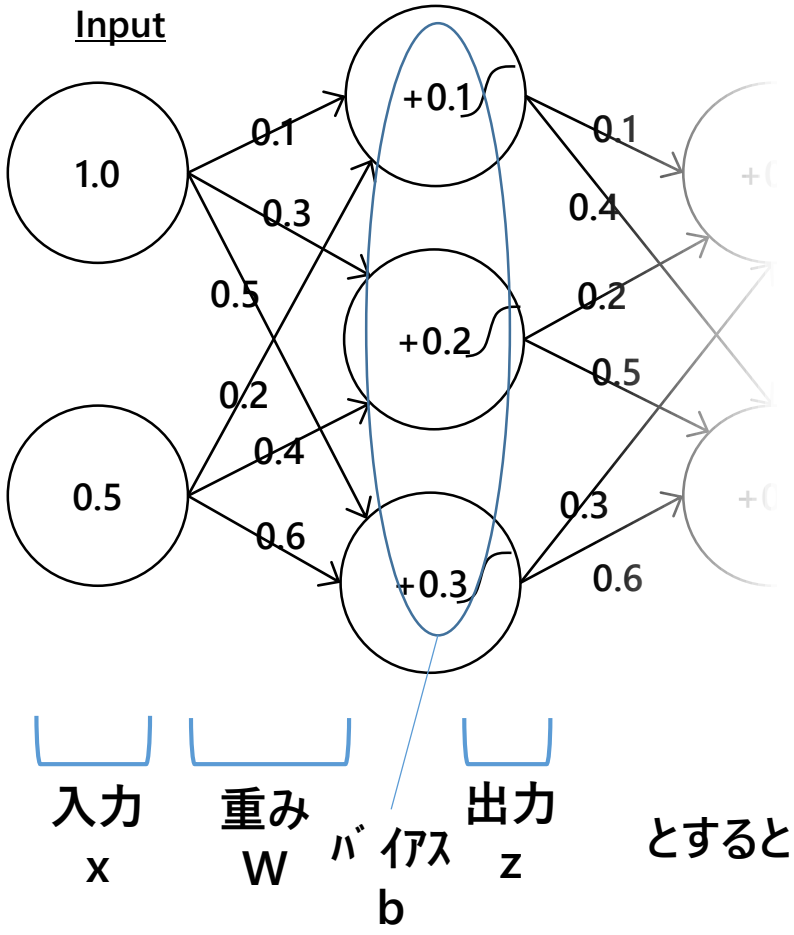
ふたたび写経する

https://github.com/yoshihiroo/programming-workshop/blob/master/deep_learning_jupyter/file1_vector_and_matrix.ipynb

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、

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

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

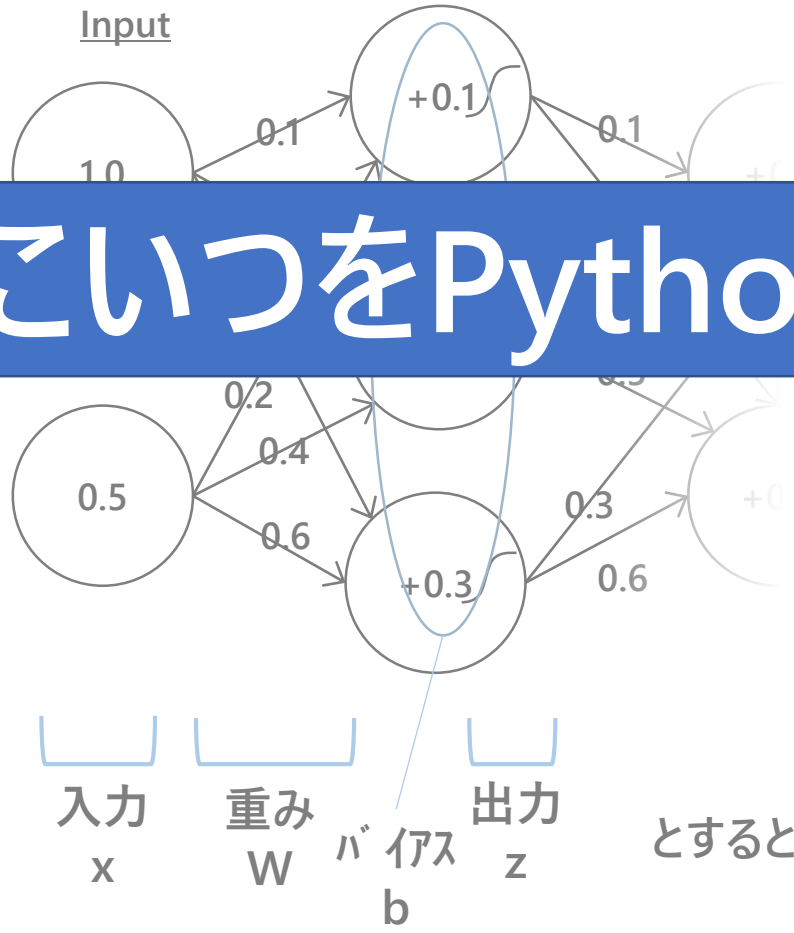
$$= \mathrm{sig}\left(\begin{pmatrix} 0.2 \\ 0.5 \\ 0.8 \end{pmatrix} + \begin{pmatrix} 0.1 \\ 0.2 \\ 0.3 \end{pmatrix}\right)$$

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

ニューラルネットワークを行列計算で表現してみる

sig() …シグモイド関数

先ほどの模型の前半部分



とすると、

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

$$= \mathrm{sig}\left(\begin{pmatrix}0.1 & 0.2\\ \cdots & \cdots\end{pmatrix}\begin{pmatrix}1.0\\ \cdots\end{pmatrix} + \begin{pmatrix}0.1\\ \cdots\end{pmatrix}\right)$$

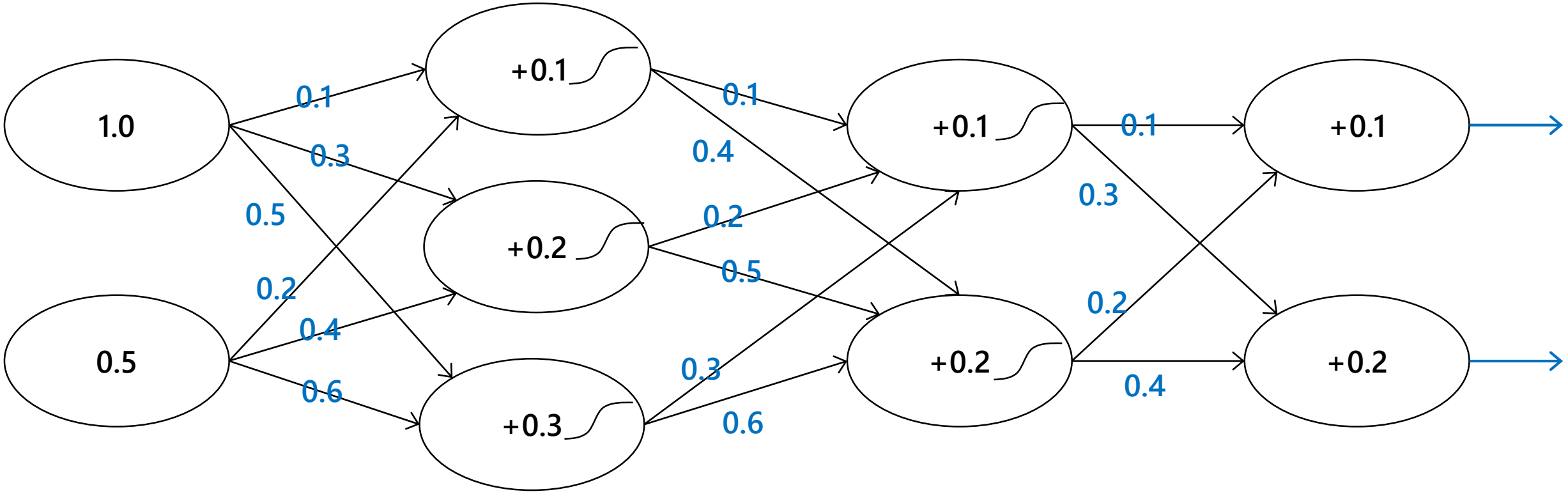
$$= \mathrm{sig}\left(\begin{pmatrix}0.2\\ 0.5\\ 0.8\end{pmatrix} + \begin{pmatrix}0.1\\ 0.2\\ 0.3\end{pmatrix}\right)$$

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

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

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

W

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

b3

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

z3

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

z1=sig(W1・x+b1)

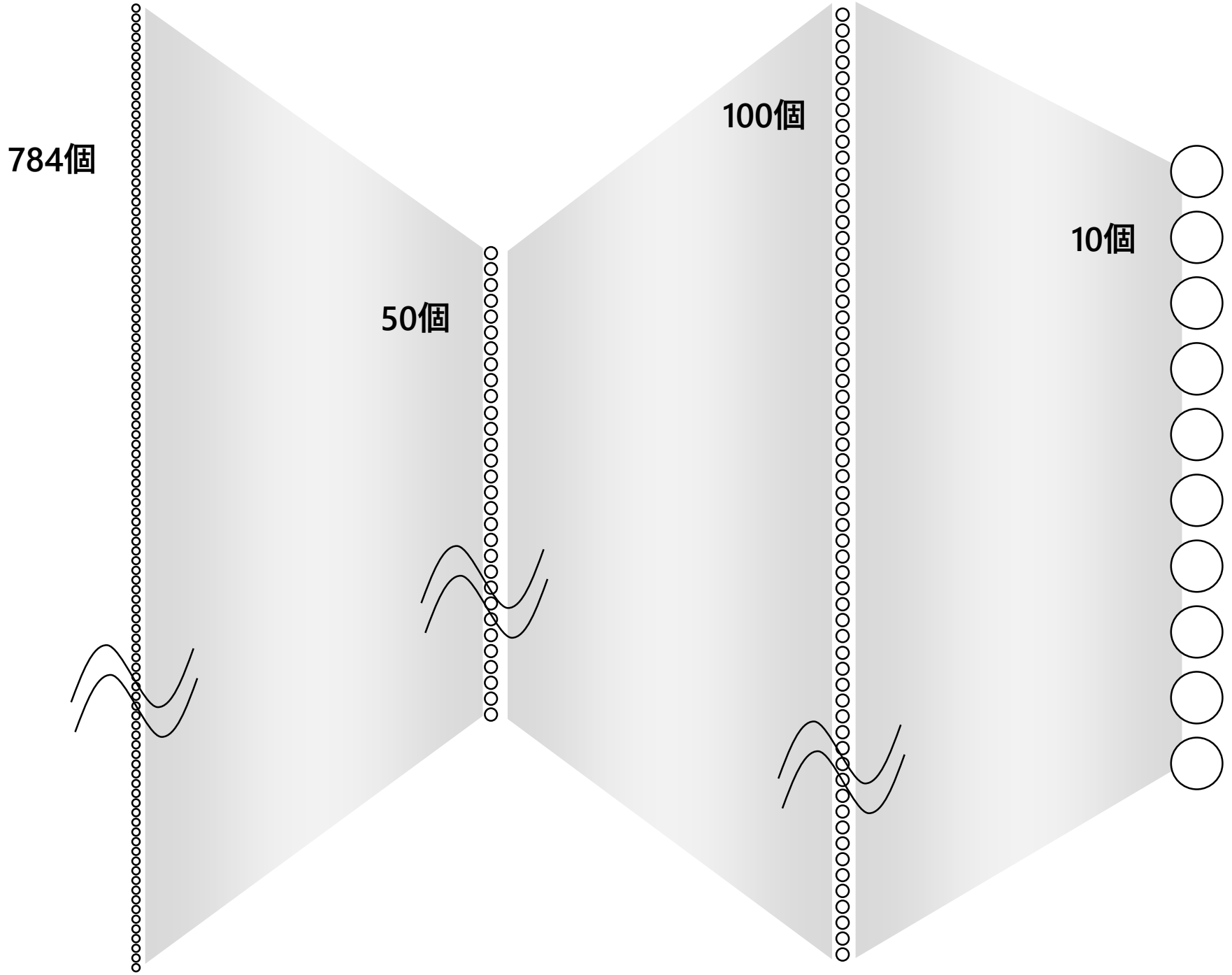
z2=sig(W2・z1+b2)

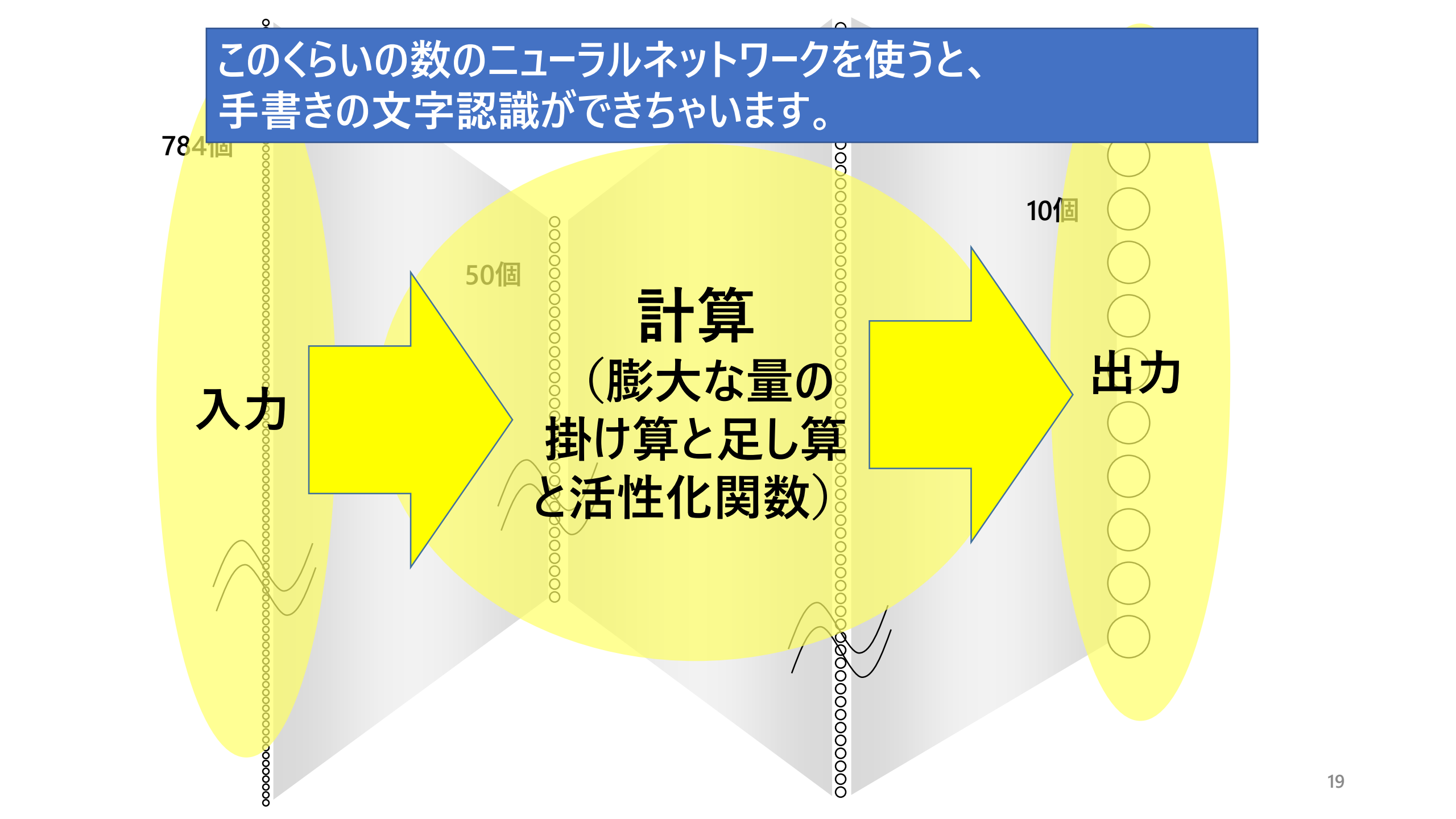
z3=W3・z2+b3

sig() …シグモイド関数

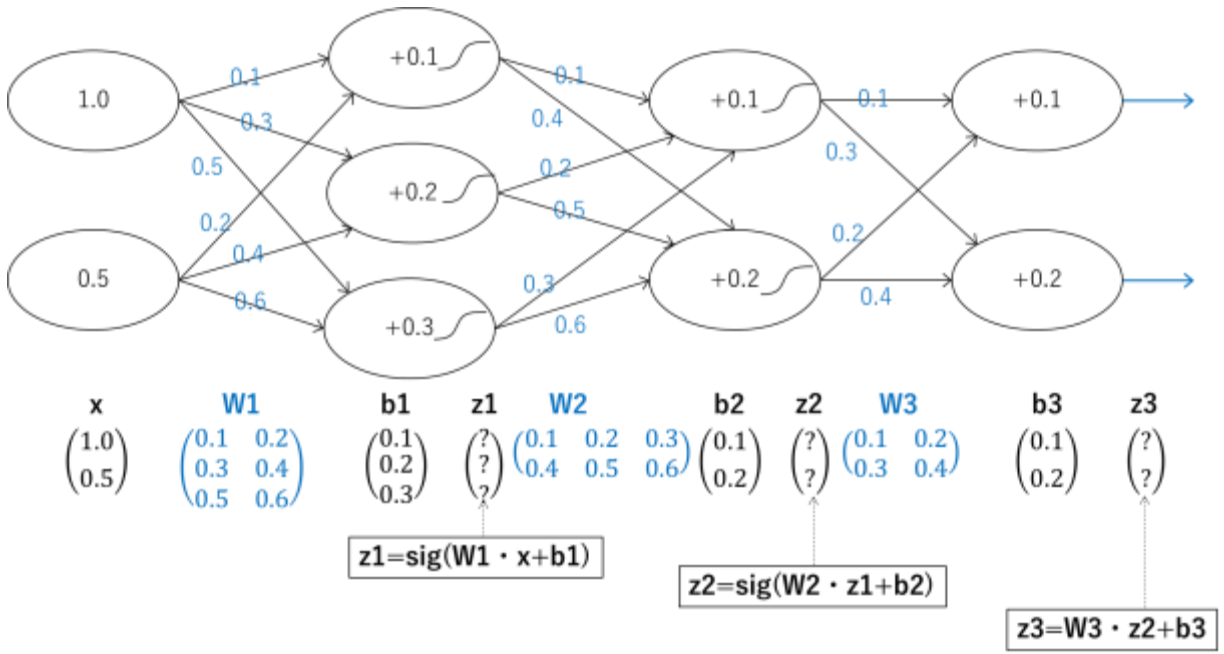
```
import numpy as np

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！

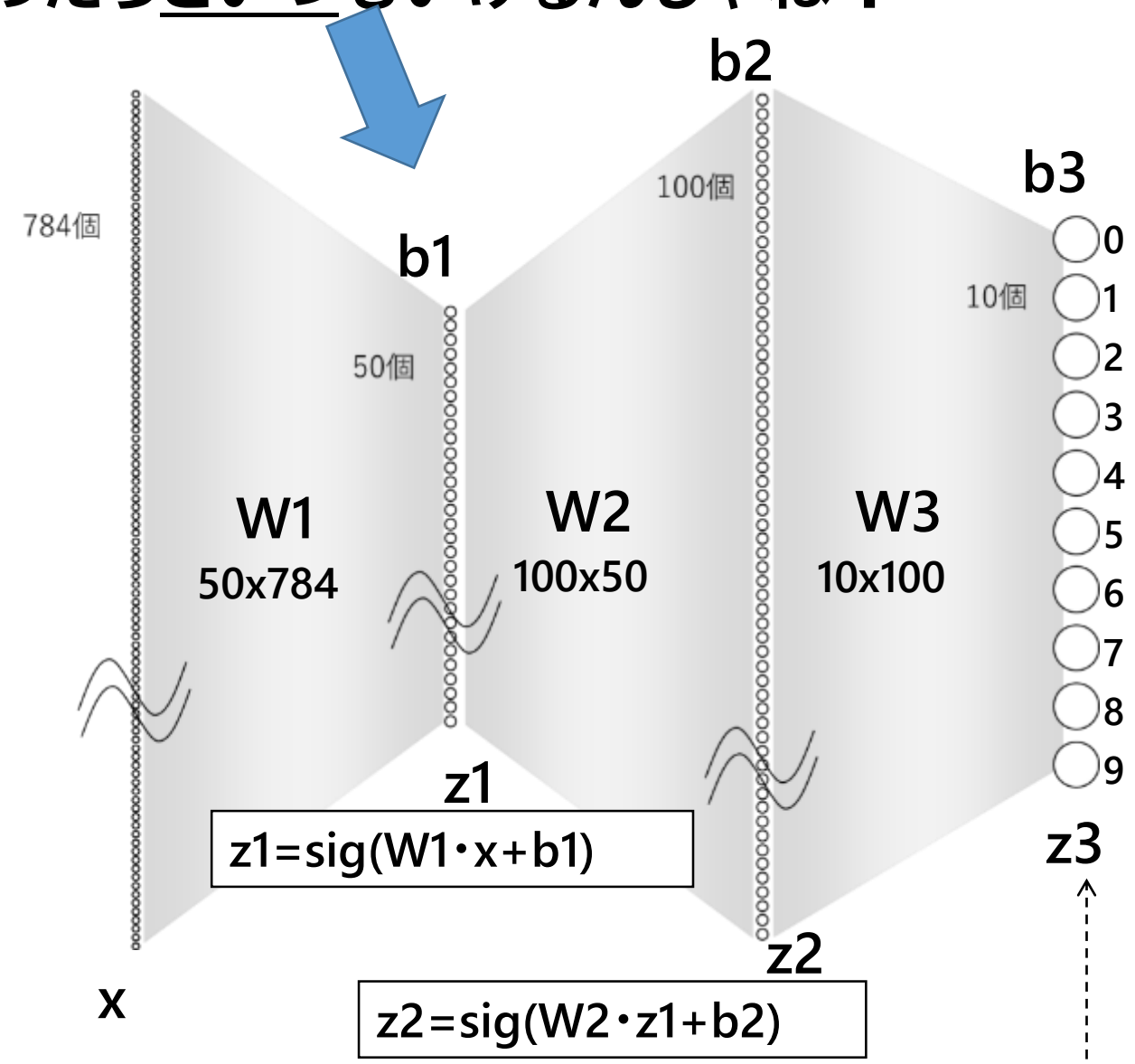

784個
50個
100個
10個


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

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。

今日やること（やれるとこまで）

- プログラミング言語「Python(パイソン)」で簡単なプログラミングを打ち込んで動かしてみる

- ディープラーニングに関連するPythonのプログラムを打ち込んで動かしてみる

- 「手書き文字認識」を題材に、ディープラーニングの学習と推論の様子を実際に動かして見てみる

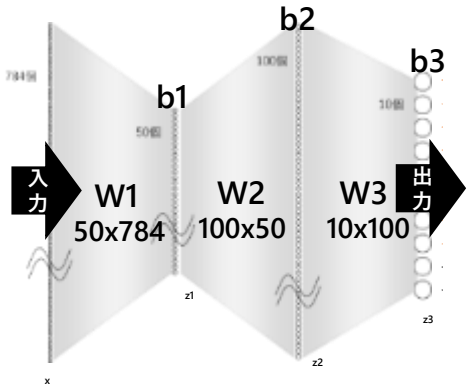
ディープラーニングの全体の流れ

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

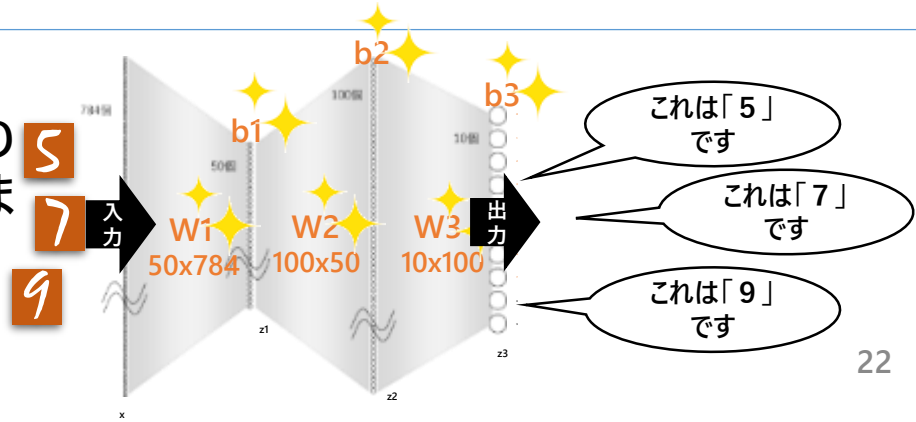
学習



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

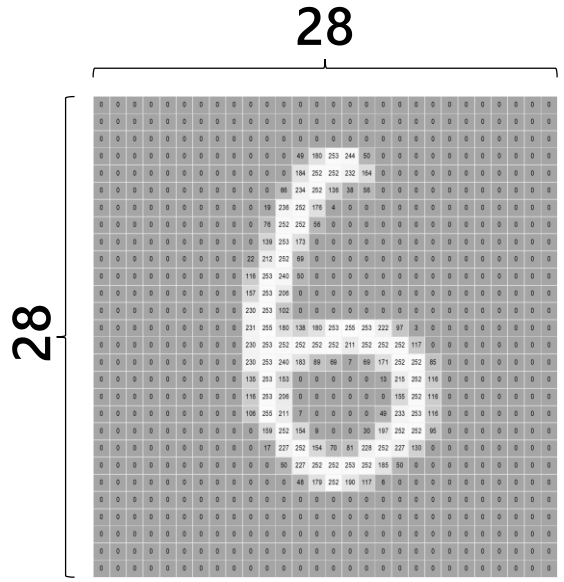
検証／適用

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



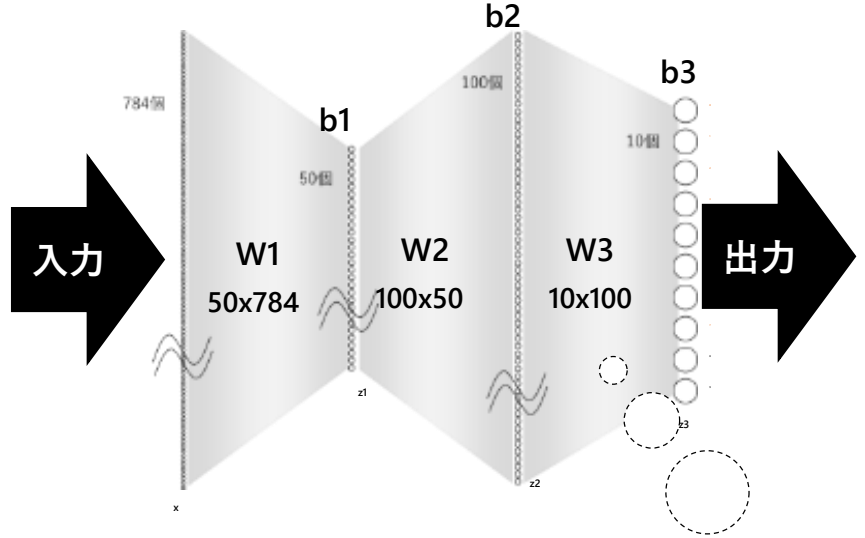
ディープラーニングにおける学習とは

手書き文字を数値化し、



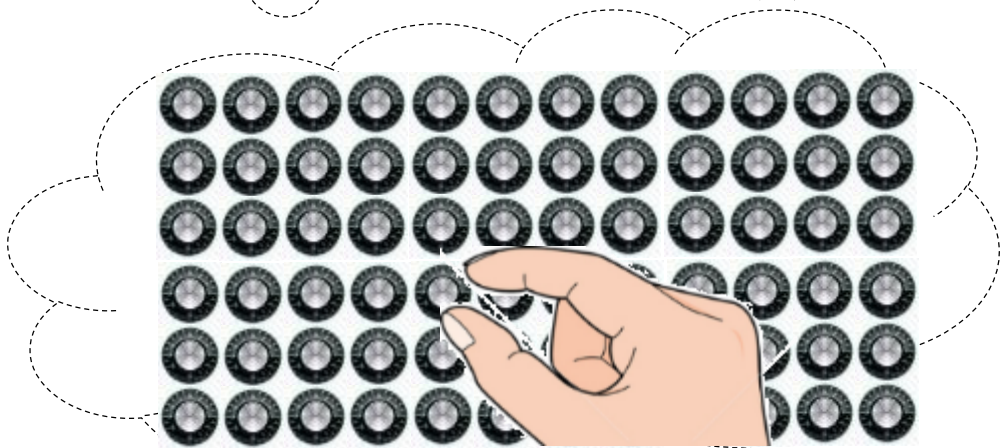
28x28=784個の格子(ピクセル)ごとに0~1の255段階の値で明るさを示すことで手書き文字を表現

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



準備

JupyterのLauncherからTerminalを開く

```
cd demo
git clone https://github.com/oreilly-japan/deep-learning-from-scratch.git
```

ch0xのディレクトリを開いてPython3のノートブックを作成する

参照するドキュメント

https://github.com/yoshihiroo/programming-workshop/blob/master/deep_learning_jupyter/file3_gakusyu.ipynb

おすすめの参考書

- 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