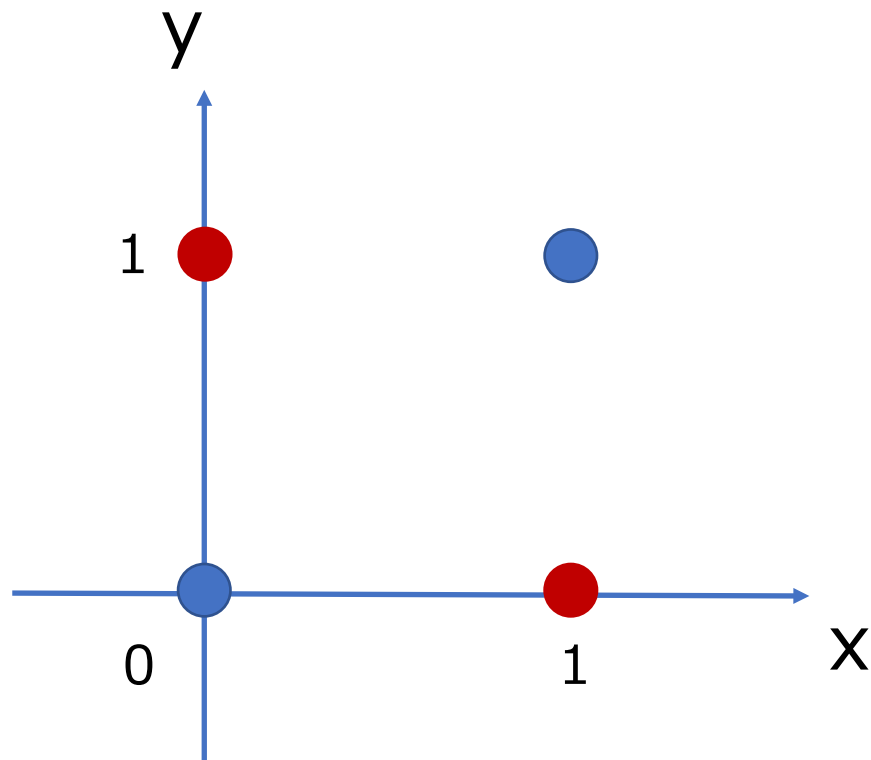


全くのゼロからのディープラーニングハンズオン

Github: [sehips](#)

ニューラルネットワークのしくみ



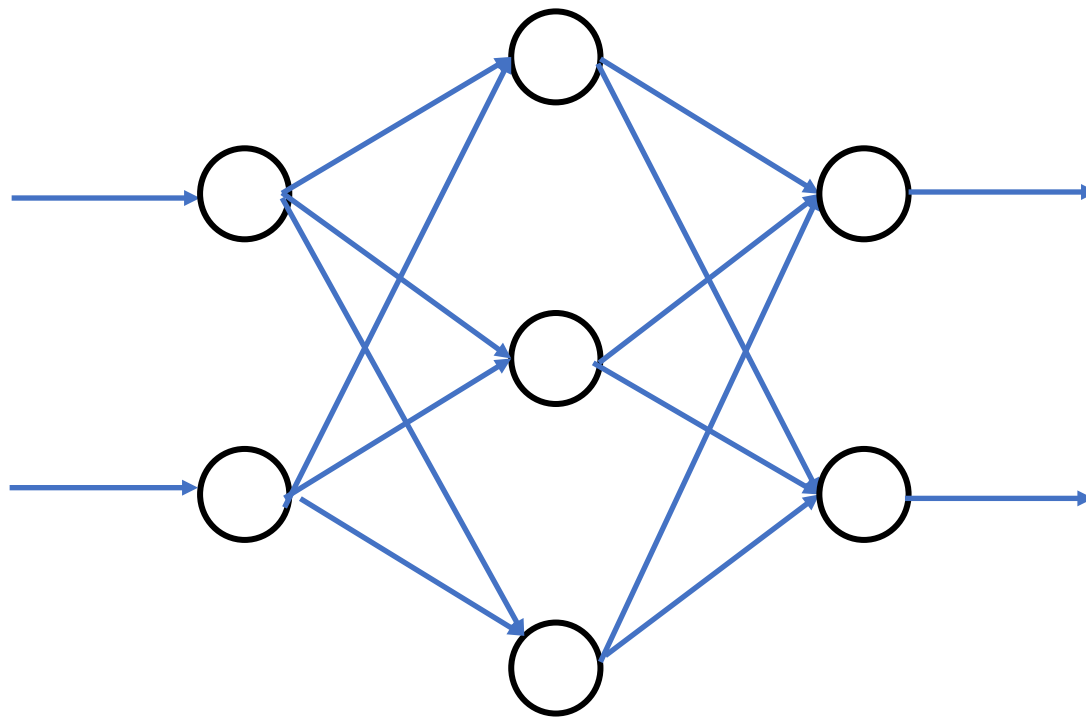
$(x, y) = (0, 0)$
 $(x, y) = (1, 1)$ } であれば青

$(x, y) = (0, 1)$
 $(x, y) = (1, 0)$ } であれば赤

と分類できる分類器を作りたい

ニューラルネットワークのしくみ

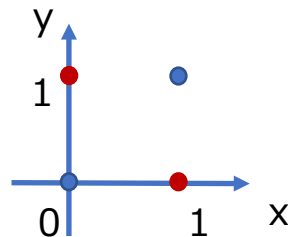
入力層が2つ、中間層が3つ、出力層が2つ
からなるニューラルネットを考える



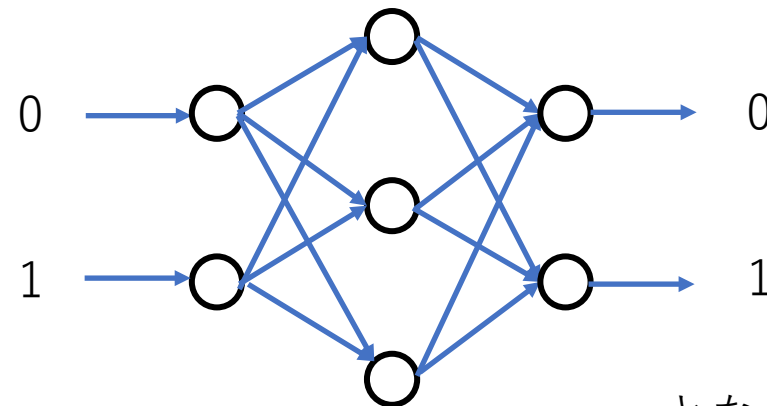
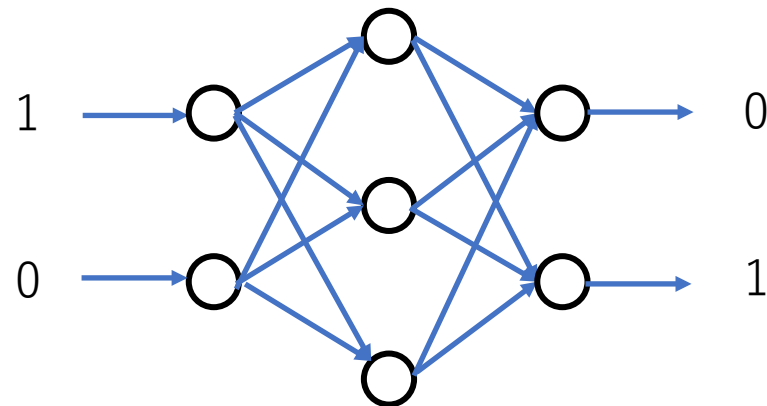
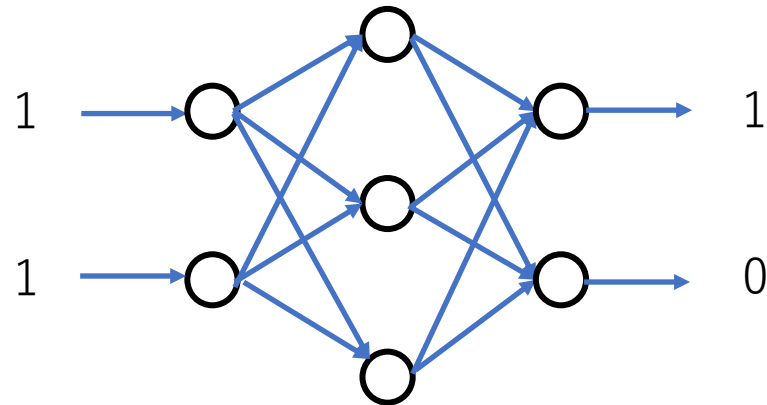
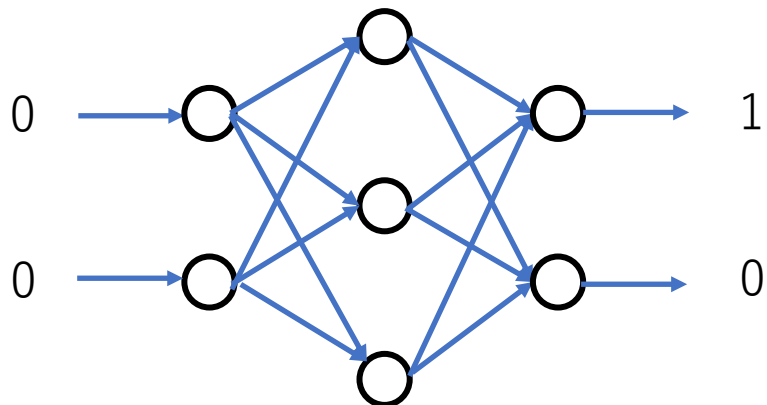
ニューラルネットワークのしくみ

$$\text{青} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{赤} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

と定義すると、

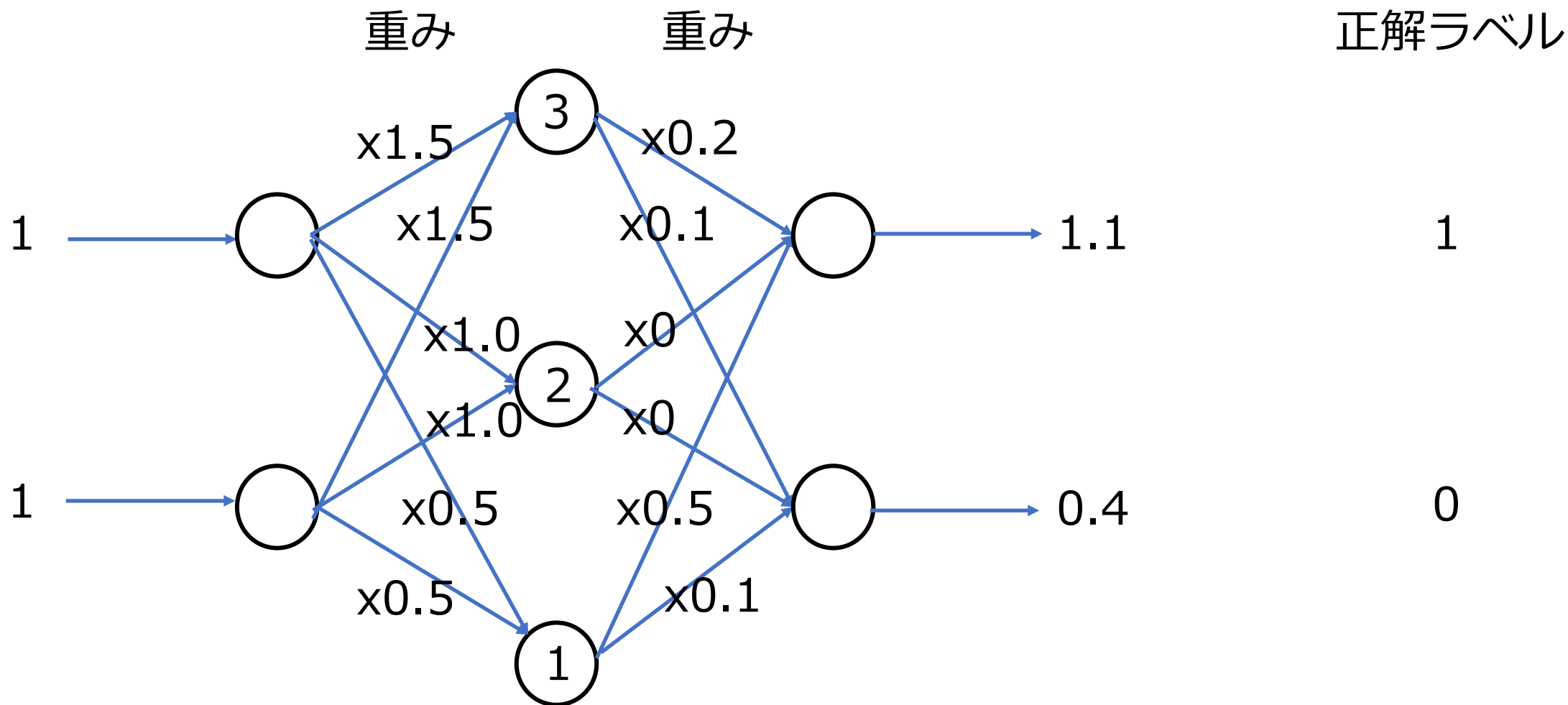


なので、



となればよい

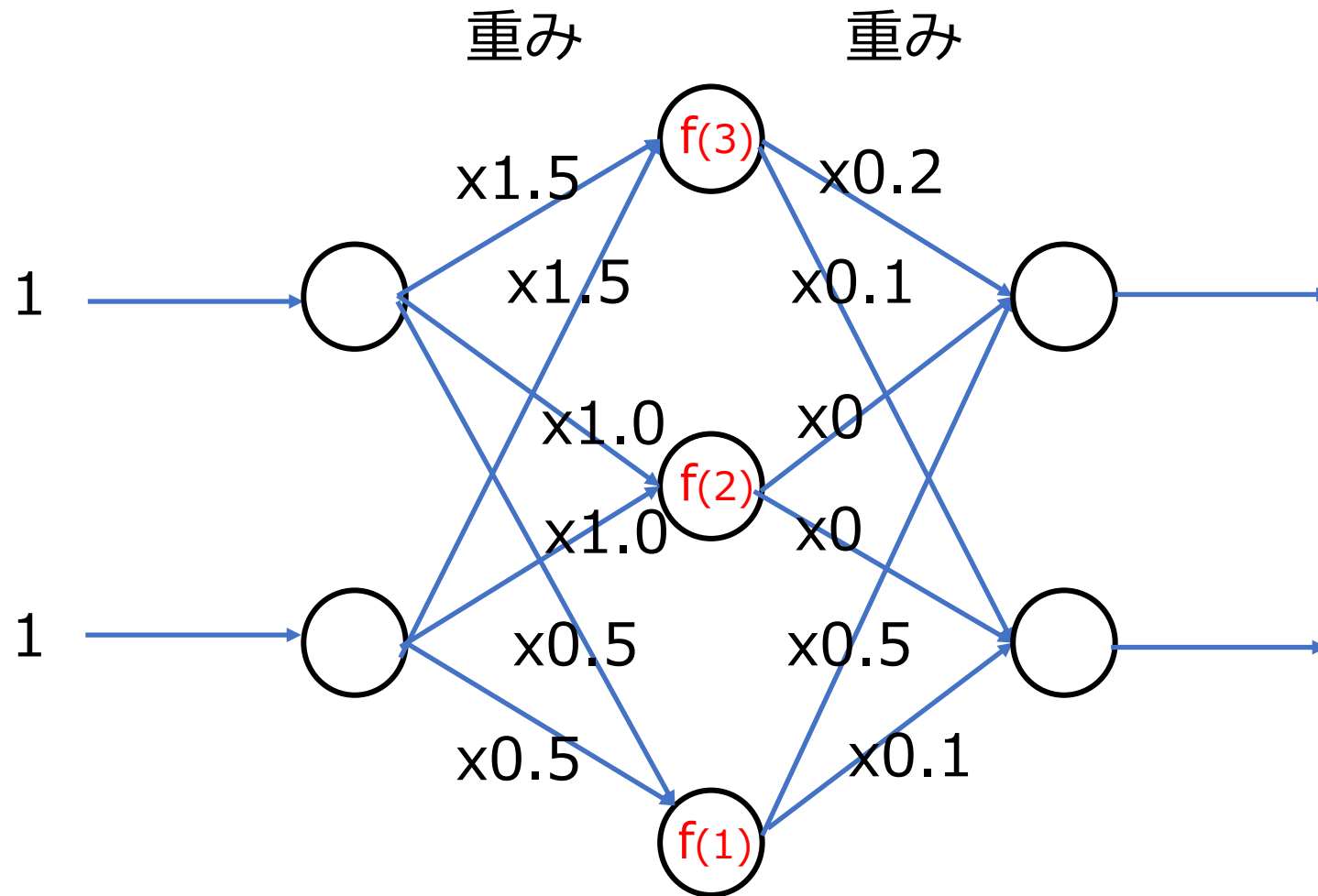
ニューラルネットワークのしくみ



誤差の二乗の和 = $(1.1 - 1)^2 + (0.4 - 0)^2 = 0.17$

誤差の二乗の和が最小になるように少しずつ重みを増減させていく → 最小二乗法

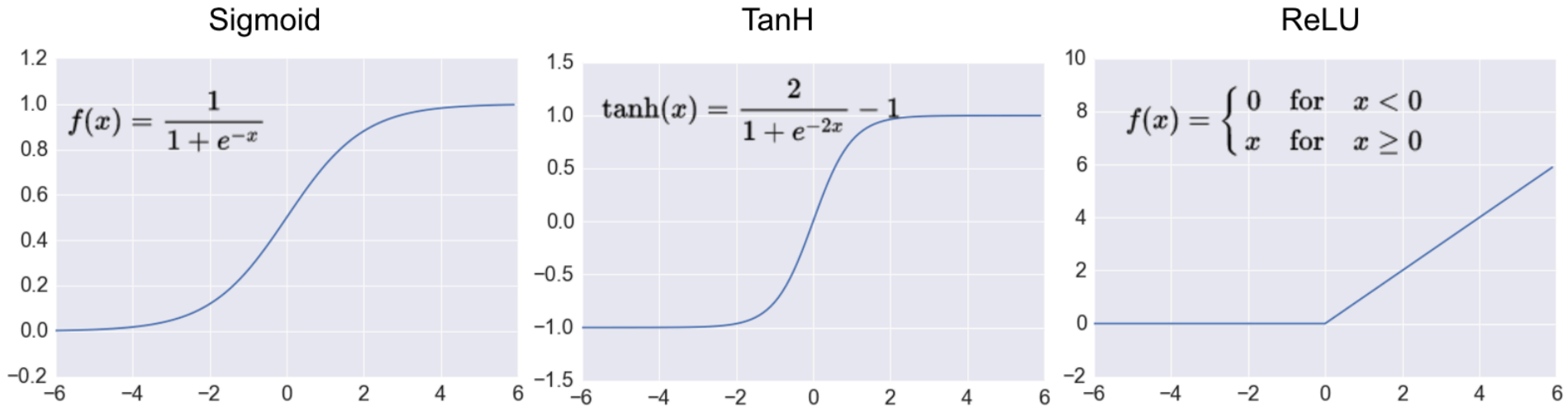
ニューラルネットワークのしくみ



実際には、中間層のニューロンは入って来た数字の合計をそのまま次のニューロンに渡すのではなく、活性化関数($f(x)$)で変換して次のニューロンに渡す。

ニューラルネットワークのしくみ

活性化関数として使われる主な関数

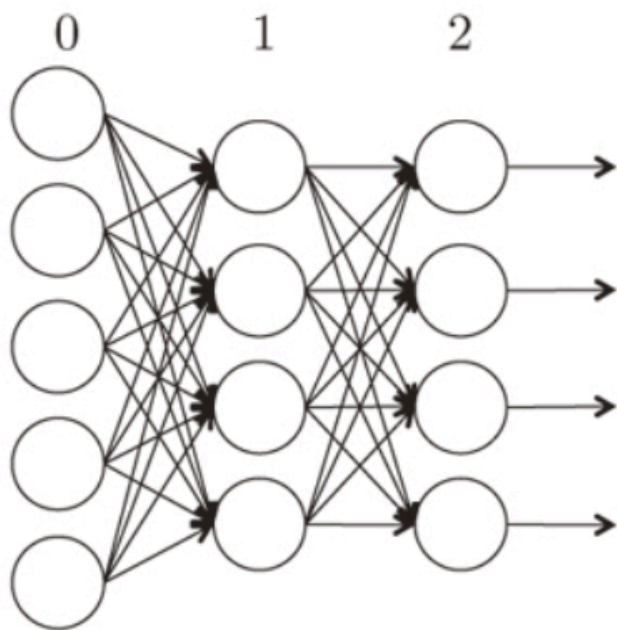


<http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/>

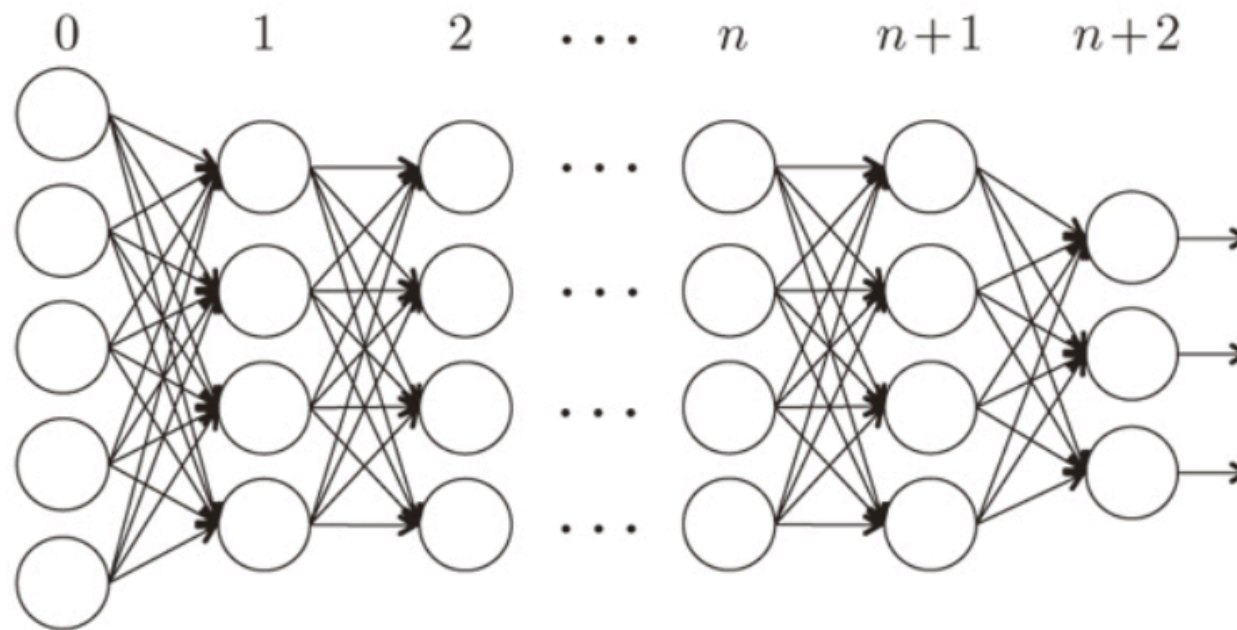
これらのような非線形関数を介在させることで、線形関数だけでは不可能な分離が可能になる。

線形関数の足し算は結局線形関数になってしまうので、活性化関数は必ず非線形関数であることが必要。

ニューラルネットワークのしくみ



浅いネットワーク



深い（ディープな）ネットワーク

ディープラーニング用環境構築



プログラミング言語の一つ

素人が書いてもプロが書いても同じような文になる

初心者が始めやすいため人気

機械学習分野に広まっている

ディープラーニング用環境構築



Python2系とPython3系がある

2系と3系は互換性がない

これから始めるなら3系がよい

2系を3系に書き直したりするのはそれほど苦ではない

ディープラーニング用環境構築



[Anaconda Cloud](#) [Documentation](#) [Blog](#) [Contact](#) [Q](#) [DOWNLOAD](#)

[What Is Anaconda?](#) [Products](#) [Support & Solutions](#) [Community](#) [About](#) [Resources](#)

**DOWNLOAD
ANACONDA NOW**

Download for



GET SUPERPOWERS WITH **ANACONDA**

Anaconda is the leading open data science platform powered by Python. The open source version of Anaconda is a high performance distribution of Python and R and includes over 100 of the most popular Python, R and Scala packages for data science.

Which version should I download and install?

With Anaconda you can run multiple versions of Python in isolated environments, so choose the download with the Python version that you use more often, as that will be your default Python version.

If you don't have time or disk space for the entire

ディープラーニング用環境構築 for Windows

Download for Windows

Download for macOS

Download for Linux

Anaconda 4.4.0 For Windows

Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.

[Changelog](#)

1. Download the installer
2. Optional: Verify data integrity with [MD5 or SHA-256](#) [More info](#)
3. Double-click the **.exe** file to install Anaconda and follow the instructions on the screen

Behind a firewall? Use these [zipped Windows installers](#)

Python 3.6 version

64-BIT INSTALLER (437M)

32-BIT INSTALLER (362M)

Python 2.7 version

64-BIT INSTALLER (430M)

32-BIT INSTALLER (354M)

ディープラーニング用環境構築

Windows

Pythonは入っていない



Python3系の環境をインストール

Mac/Linux

Python2系が元から入っている



Python3系を入れて元からある
Python2系が消えてしまうとい
ろいろ心配

Windows用：ディープラーニング用環境構築

<https://repo.continuum.io/archive/>

| | | | |
|------------------------------------|--------|---------------------|----------------------------------|
| Anaconda2-4.2.0-MacOSX-x86_64.pkg | 403.9M | 2016-10-17 19:33:11 | cd2ccc991b7f1503335367d80d0317b0 |
| Anaconda3-4.2.0-MacOSX-x86_64.pkg | 407.1M | 2016-10-17 19:33:47 | 51ed7f9af7436a1a23068eb00509d6ad |
| Anaconda2-4.2.0-Linux-x86.sh | 365.0M | 2016-09-27 15:50:20 | e26582ebdf1d982e18efb2bdf52c5ee6 |
| Anaconda2-4.2.0-Linux-x86_64.sh | 446.0M | 2016-09-27 15:49:54 | a0d1fbe47014b71c6764d76fb403f217 |
| Anaconda2-4.2.0-MacOSX-x86_64.sh | 346.4M | 2016-09-27 15:50:02 | 52f8b74e0c462575efc297c8f4e6cf14 |
| Anaconda2-4.2.0-Windows-x86.exe | 324.1M | 2016-09-27 15:54:50 | f4f12af8811759e56464eef5a484963d |
| Anaconda2-4.2.0-Windows-x86_64.exe | 381.0M | 2016-09-27 15:55:47 | 0a30d509568724dac0ae193e139b9c37 |
| Anaconda3-4.2.0-Linux-x86.sh | 373.9M | 2016-09-27 15:50:34 | 7aca10e1ea5b9db0a318b4eed5253747 |
| Anaconda3-4.2.0-Linux-x86_64.sh | 455.9M | 2016-09-27 15:50:04 | 4692f716c82deb9fa6b59d78f9f6e85c |
| Anaconda3-4.2.0-MacOSX-x86_64.sh | 349.5M | 2016-09-27 15:50:07 | 7cb61e355eb860e342a5e27236e3f375 |
| Anaconda3-4.2.0-Windows-x86.exe | 333.4M | 2016-09-27 15:56:30 | 96e5fe052b22d667da9360fb4edce363 |
| Anaconda3-4.2.0-Windows-x86_64.exe | 391.4M | 2016-09-27 15:57:21 | 0ca5ef4dcfe84376aad073bbb3f8db00 |
| Anaconda2-4.1.1-Linux-x86.sh | 324.6M | 2016-07-08 11:19:57 | 8813071788e08e236a323b5f7d337759 |
| Anaconda2-4.1.1-Linux-x86_64.sh | 399.6M | 2016-07-08 11:19:56 | f7bb3c0ccf23c9789bb895335aa68bf3 |
| Anaconda2-4.1.1-MacOSX-x86_64.pkg | 345.0M | 2016-07-08 11:19:59 | e88beae19868dc01fae908dd1e067bda |
| Anaconda2-4.1.1-MacOSX-x86_64.sh | 295.8M | 2016-07-08 11:20:00 | f62a0a47a42504e139a5122ad641b40c |
| Anaconda2-4.1.1-Windows-x86.exe | 286.0M | 2016-07-08 11:20:01 | b319d6867c67723ba74aef4f9dd35f82 |
| Anaconda2-4.1.1-Windows-x86_64.exe | 341.2M | 2016-07-08 11:20:01 | 1db0244dbf02579f452d1b19ce245144 |
| Anaconda3-4.1.1-Linux-x86.sh | 329.1M | 2016-07-08 11:20:02 | 0576a0df8987ca62d5c13491102547d9 |
| Anaconda3-4.1.1-Linux-x86_64.sh | 406.3M | 2016-07-08 11:20:02 | d0dc08d241f83ffc763504db50008e5b |
| Anaconda3-4.1.1-MacOSX-x86_64.pkg | 347.9M | 2016-07-08 11:21:15 | 9d396421683249ae850bd19637577f6e |
| Anaconda3-4.1.1-MacOSX-x86_64.sh | 298.7M | 2016-07-08 11:21:17 | 185aa68d5841869cb7cb3a031bd63936 |
| Anaconda3-4.1.1-Windows-x86.exe | 293.8M | 2016-07-08 11:21:18 | 39bd047c2169a9d072e98403f487c9e8 |
| Anaconda3-4.1.1-Windows-x86_64.exe | 352.9M | 2016-07-08 11:21:17 | a3be394f8274c391148efdfbc63e8ca4 |

最新版のPython3.6が入ったAnacondaはTensorflowに対応していないらしいので（その後対応したかもしれないが未確認なので）、Windowsの人は上記のリンクからAnaconda3-4.2.0をダウンロードし、インストールをお願いします。

ディープラーニング用環境構築 for Mac/Linux

元からあるPython2系を残したままPython3系のAnacondaを入れるために pyenv というバージョン管理ツールを利用する（以下はターミナル上で実行）

#pyenvのinstall

```
curl -L https://raw.githubusercontent.com/pyenv/pyenv-installer/master/bin/pyenv-installer | bash
echo 'export PATH="~/.pyenv/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(pyenv init -)"' >> ~/.bashrc
echo 'eval "$(pyenv virtualenv-init -)"' >> ~/.bashrc
source ~/.bashrc
```

#Anacondaのinstall

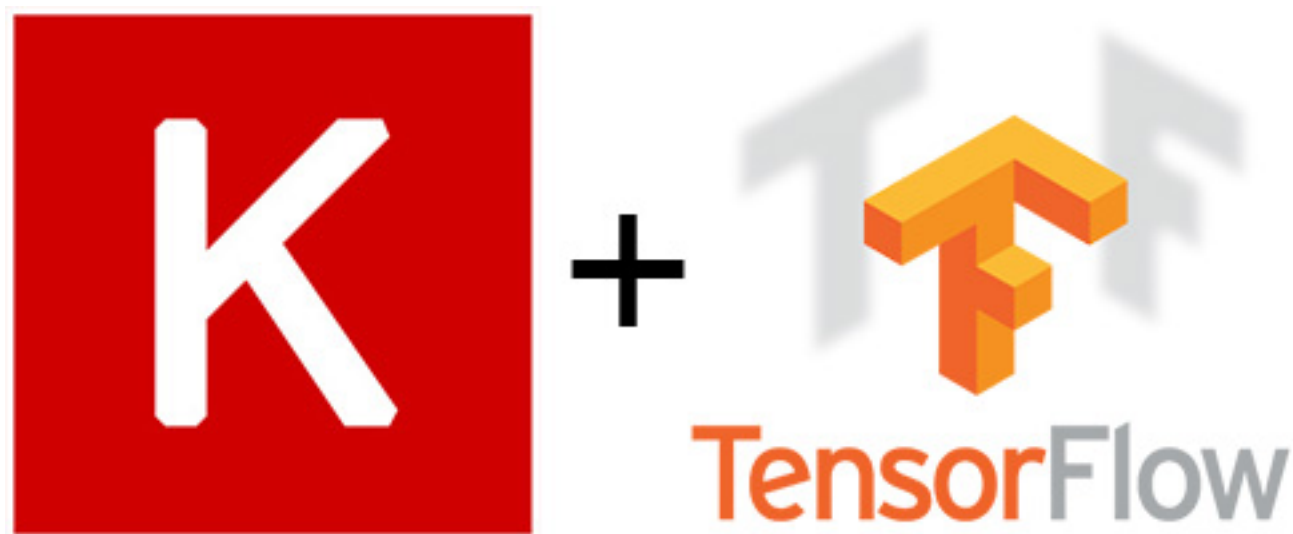
```
pyenv install anaconda3-4.2.0
pyenv global anaconda3-4.2.0
echo 'export PATH="$PYENV_ROOT/versions/anaconda3-4.2.0/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
```

同じく最新版のPython3.6が入ったAnacondaはTensorflowに対応していないらしいので（その後対応したかもしれないが未確認なので）、Mac/Linuxの人は上記のコマンドをターミナルにコピーして Anaconda3-4.2.0をダウンロードし、インストールをお願いします。

代表的な深層学習用フレームワーク

- Tensorflow Googleが開発
- Caffe カリフォルニア大学のバークレー校の研究センターが開発
- Torch NEC北米研究所が開発 現在はFacebookが開発に関わっている
- CNTK マイクロソフトが開発
- Chainer Preferred Networks（日本のベンチャー企業）が開発

おそらく最も簡単なので、KerasとTensorflowを使う





TensorFlow, CNTK, Theanoをバックエンドとして動作し, 使いやすく抽象化された環境を提供するニューラルネットワーク用のライブラリ

作者はGoogleのエンジニア
なぜが日本語が話せる

公式ドキュメントの日本語版が存在する。
(Tensorflowは英語のみ)



<https://keras.io/ja/>

要はKerasを使うとTensorflowが簡単に操れる

ディープラーニング用環境構築 for Windows/Mac/Linux

Windowsではコマンドプロンプト、Mac/Linuxではターミナルを開き、以下を入力

```
pip install tensorflow  
pip install keras
```

* GPUを使う場合にはpip install tensorflow-gpu

jupyterを開くために以下を入力

```
jupyter notebook
```

Jupyterが開いたら右上のNewをクリックしてPythonを選択



Logout

Files

Running

Clusters

Conda

Select items to perform actions on them.

Upload

New ▾



Text File

Folder

Terminal

Notebooks

Python [Root]

R

opencvtest



☐ 20170313

☐ anaconda

☐ Analysis_with_seurat

☐ Applications

☐ CellDesignerSim

☐ chainer-handson

☐ chap2

☐ Creative Cloud Files

☐ datascience_meeting_20170212

☐ deel

☐ deep-learning-from-scratch



In []: |

ここにコードを入力してここを押すと、入力したプログラムが実行される
(* Shift+Enterでもプログラムが実行される)

Pythonを使った簡単な数値計算

```
In [1]: 1+1
```

```
Out[1]: 2
```

```
In [2]: 2*2
```

```
Out[2]: 4
```

```
In [3]: 2/2
```

```
Out[3]: 1.0
```

```
In [4]: 4-5
```

```
Out[4]: -1
```

```
In [5]: 4**5
```

```
Out[5]: 1024
```

```
In [6]: a = 5**4
```

```
In [7]: a
```

```
Out[7]: 625
```

```
In [8]: a = 6
```

```
In [9]: a
```

```
Out[9]: 6
```

```
In [10]: a/6
```

```
Out[10]: 1.0
```

```
In [11]: a = [1,3,4,8]
```

```
In [12]: a
```

```
Out[12]: [1, 3, 4, 8]
```

```
In [15]: a*2
```

```
Out[15]: [1, 3, 4, 8, 1, 3, 4, 8]
```

```
In [19]: a.append(2)
```

```
In [20]: a
```

```
Out[20]: [1, 3, 4, 8, 2]
```

Numpyを使うと多くの数字の計算を同時に扱うことができる

```
In [22]: import numpy as np
```

```
In [25]: a = np.array(a)
```

```
In [26]: a
```

```
Out[26]: array([1, 3, 4, 8, 2])
```

```
In [27]: a*2
```

```
Out[27]: array([ 2,  6,  8, 16,  4])
```

```
In [28]: a-1
```

```
Out[28]: array([0, 2, 3, 7, 1])
```

```
In [29]: a**2
```

```
Out[29]: array([ 1,  9, 16, 64,  4])
```

```
In [30]: a+a
```

```
Out[30]: array([ 2,  6,  8, 16,  4])
```

```
In [31]: a*a
```

```
Out[31]: array([ 1,  9, 16, 64,  4])
```

```
In [33]: b = np.array([[1,2,3],[4,5,6],[7,8,9]])
```

```
In [34]: b
```

```
Out[34]: array([[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]])
```

```
In [35]: c = np.array([[2,2,4],[4,4,6],[8,8,6]])
```

```
In [36]: c
```

```
Out[36]: array([[2, 2, 4],
               [4, 4, 6],
               [8, 8, 6]])
```

```
In [37]: b+c
```

```
Out[37]: array([[ 3,  4,  7],
               [ 8,  9, 12],
               [15, 16, 15]])
```

```
In [38]: b*c
```

```
Out[38]: array([[ 2,  4, 12],
               [16, 20, 36],
               [56, 64, 54]])
```

```
In [39]: b.shape
```

```
Out[39]: (3, 3)
```

```
In [41]: d = np.array([b,b,b,b,c,c])
```

```
In [42]: d
```

```
Out[42]: array([[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]],
               [[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]],
               [[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]],
               [[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]],
               [[2, 2, 4],
               [4, 4, 6],
               [8, 8, 6]],
               [[2, 2, 4],
               [4, 4, 6],
               [8, 8, 6]])
```

```
In [43]: d.shape
```

```
Out[43]: (6, 3, 3)
```


Numpyトレーニング

100 numpy exercises

A joint effort of the numpy community

The goal is both to offer a quick reference for new and old users and to provide also a set of exercises for those who teach. If you remember having asked or answered a (short) problem, you can send a pull request. The format is:

```
#. Find indices of non-zero elements from [1,2,0,0,4,0]

.. code:: python

# Author: Somebody

print(np.nonzero([1,2,0,0,4,0]))
```

Here is what the page looks like so far: <http://www.labri.fr/perso/nrougier/teaching/numpy.100/index.html>

Repository is at: <https://github.com/rougier/numpy-100>

Thanks to Michiaki Ariga, there is now a [Julia version](#).

1. Import the numpy package under the name `np` (☆☆☆)

```
import numpy as np
```

2. Print the numpy version and the configuration (☆☆☆)

```
print(np.__version__)
np.show_config()
```

3. Create a null vector of size 10 (☆☆☆)

```
Z = np.zeros(10)
print(Z)
```

<http://www.labri.fr/perso/nrougier/teaching/numpy.100/>
<https://github.com/rougier/numpy-100>

Jupyter notebookファイル(ipynb)

1.TensorflowとKerasを用いて多層パーセプトロンでディープラーニング

TensorflowとKerasを用いてmnistという手書き文字のデータを学習させる。多層パーセプトロンから成るニューラルネットをkerasで組み、学習させて正確性を検証する。(1.mnist_with_keras.ipynb)

2.TensorflowとKerasで畳み込みニューラルネットワークでディープラーニング

TensorflowとKerasを用いてmnistという手書き文字のデータを学習させる。畳み込みニューラルネットワークをKerasで組み、学習させて正確性を検証する。(2.mnist_with_keras_conv2d.ipynb)

3.TensorflowとKerasでディープラーニング（犬と猫の2クラス認識）

TensorflowとKerasを用いてKaggleのDogs vs. Catsで提供されている犬と猫の画像のデータ(<https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition>)を学習させる。

これまでのMNISTのようにあらかじめ準備されているデータセットではなく、画像で配布されているデータセットを用いるため、自分で画像を読み込んで学習させる必要がある。(dog_or_cat_with_keras.ipynb)