# Deep Learning Seminar 1

渡邊研究室　M1 二見 悠樹

# 多用される「DeepLearning」





# はじめに

本セミナーは、**深層学習（Deep Learning）**の**理解**が目的

基礎となるニューラルネットワーク（神経回路網）から、発展した畳込みニューラルネットワーク(CNN)，時系列を扱える再帰型ネットワーク(RNN)，その他ホットなネットワークについて解説・紹介

難しい計算とかは触れるつもりはない (僕もよくわからん)

なんも知らん人向けに解説するので独学してる人にはつまらんかも...

（復習と思ってください）

セミナー予定日時：毎週水曜13時半～(仮)

まぁ、気軽に質問して下さいなw

# 予定

1. 導入・形式ニューロン/単純パーセプトロン
2. 多層パーセプトロン（MLP）/ロジスティック回帰
3. ディープニューラルネットワーク/CNN解説
4. データ整形 CNN実装
5. 大会
6. RNN解説・実装
7. ….

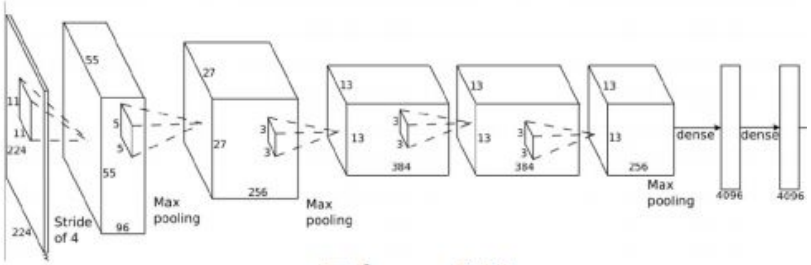
# 目次

- Deep Learning 事例
- 機械学習とは？
- ニューラルネットワークの基本
- 多層パーセプトロン
- ニューラルネットワークの実装

# Deep Learning 事例

- 画像認識・物体認識（CNN R-CNN YOLO SSD ...）
  - IMAGENET Large Scale Visual Recognition Challenge 2012
    - 1000カテゴリ・カテゴリあたり約1000枚の訓練画像

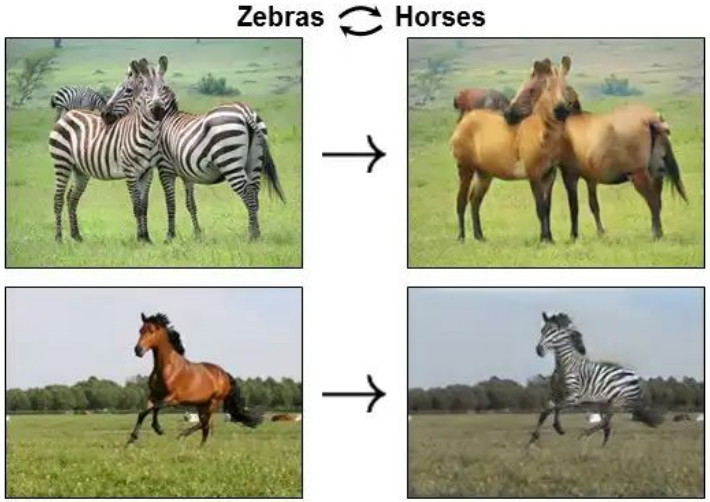

| | Team name | Error (5 guesses) |
|---|---|---|
| 1 | SuperVision | 0.15315 |
| 2 | ISI | 0.26172 |
| 3 | OXFORD_VGG | 0.26979 |
| 4 | XRCE/INRIA | 0.27058 |
| 5 | University of Amsterdam | 0.29576 |
| 6 | LEAR-XRCE | 0.34464 |

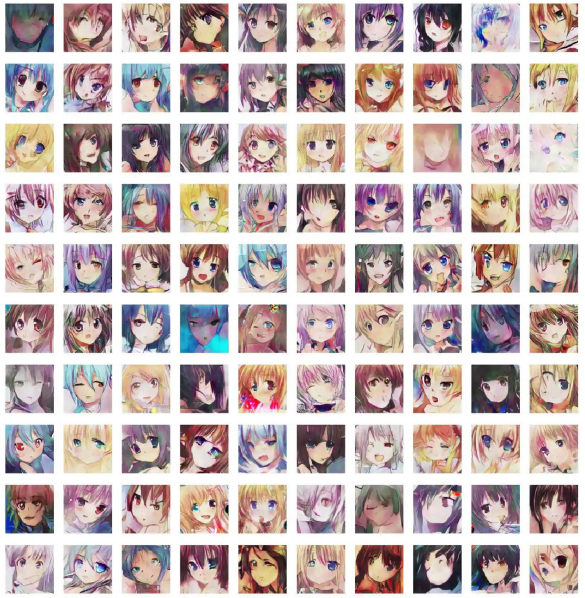


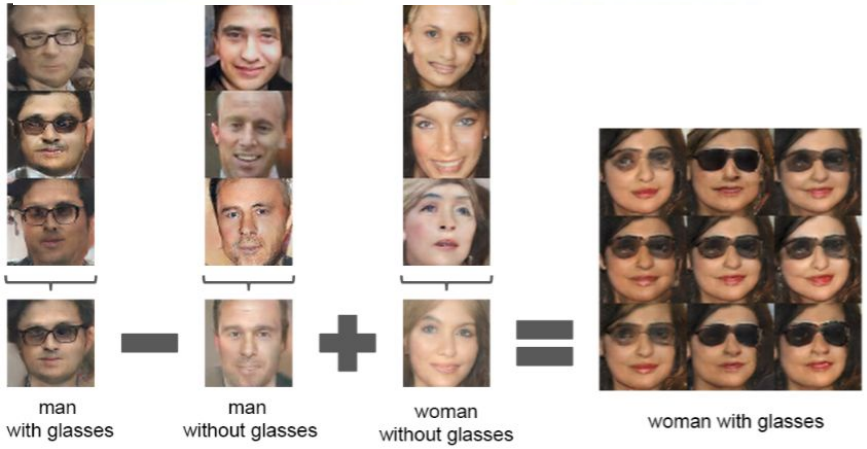

7-layer NN

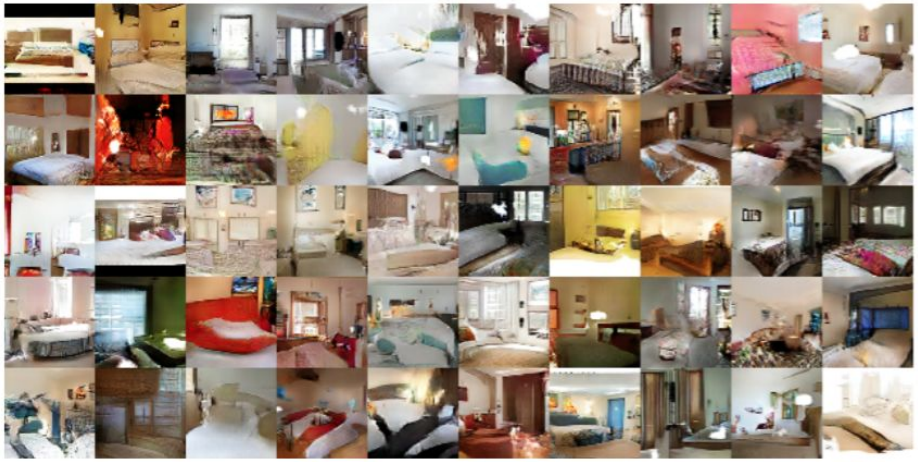
# Deep Learning事例2

画像生成 (GAN VAE)



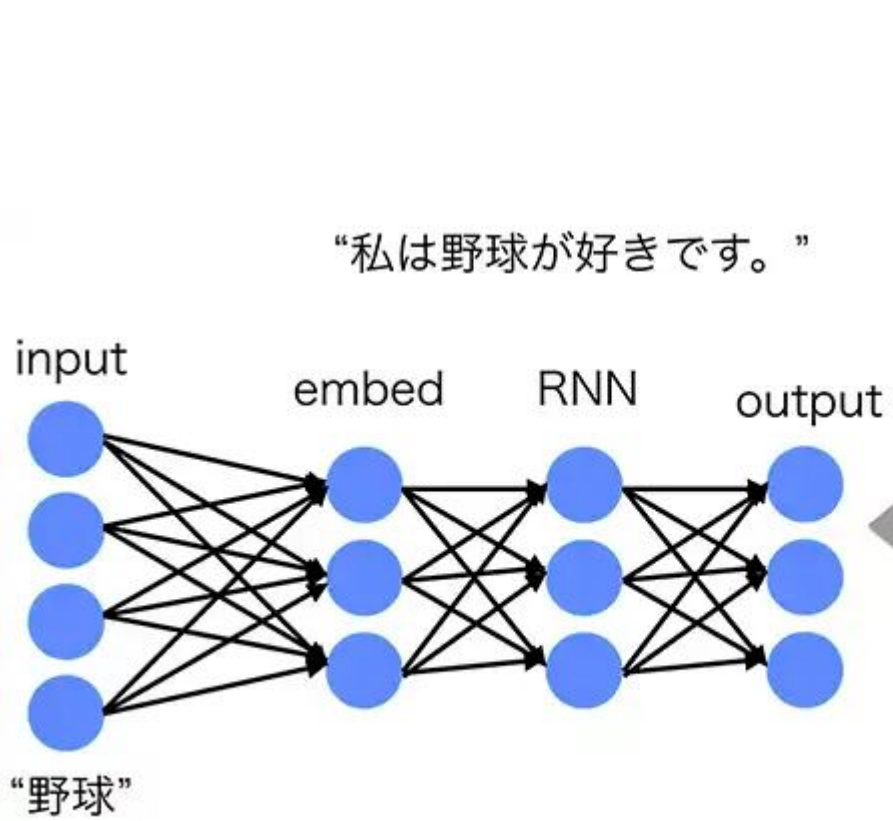


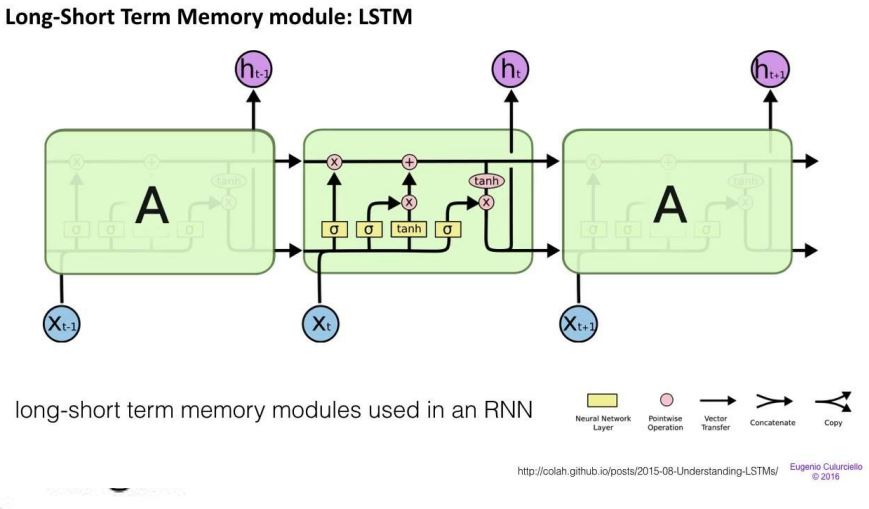



# Deep Learning事例3

- ❖ 音声認識・自動翻訳 (RNN LSTM GRU AE)
- ❖ 音声合成

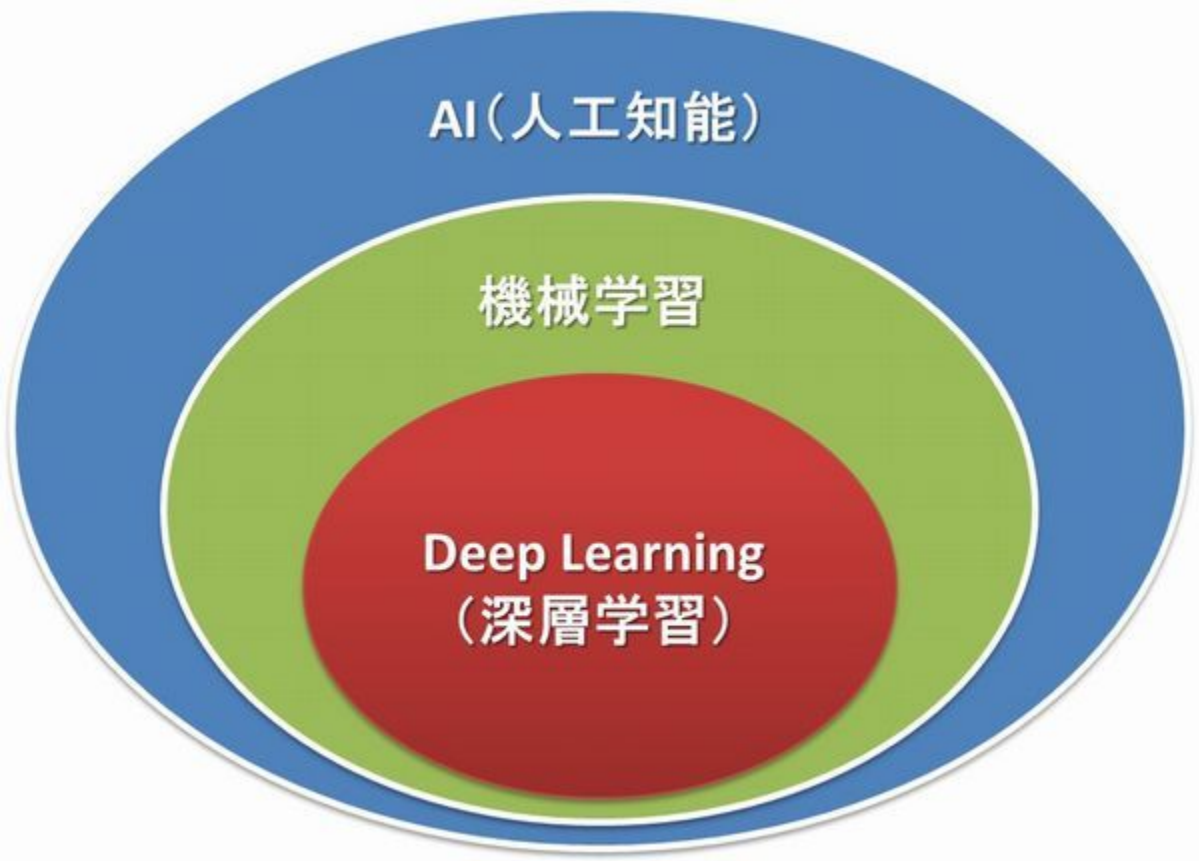
…とかもういろいろに応用されてる今注目の分野です.



# 「AIやら機械学習やらDLやら用語が混合してわからん！」

# 「AIやら機械学習やらDLやら用語が混合してわからん！」

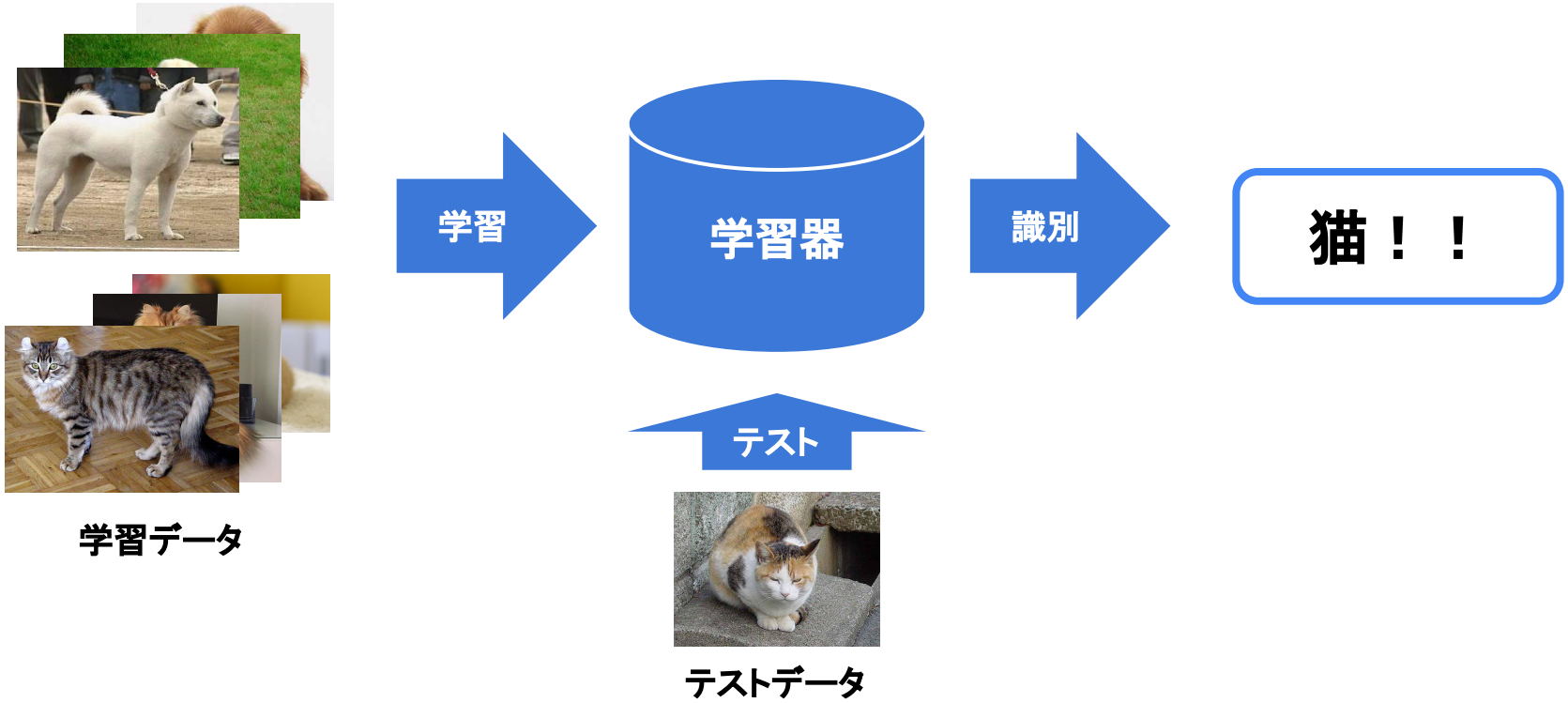


# 機械学習とは？

人間の学習能力と同等の機能をコンピュータで再現する技術

おおよその問題（データ）は**何らかのパターン**を含んでおり、それを反復的に学習して
　パターンを発見する

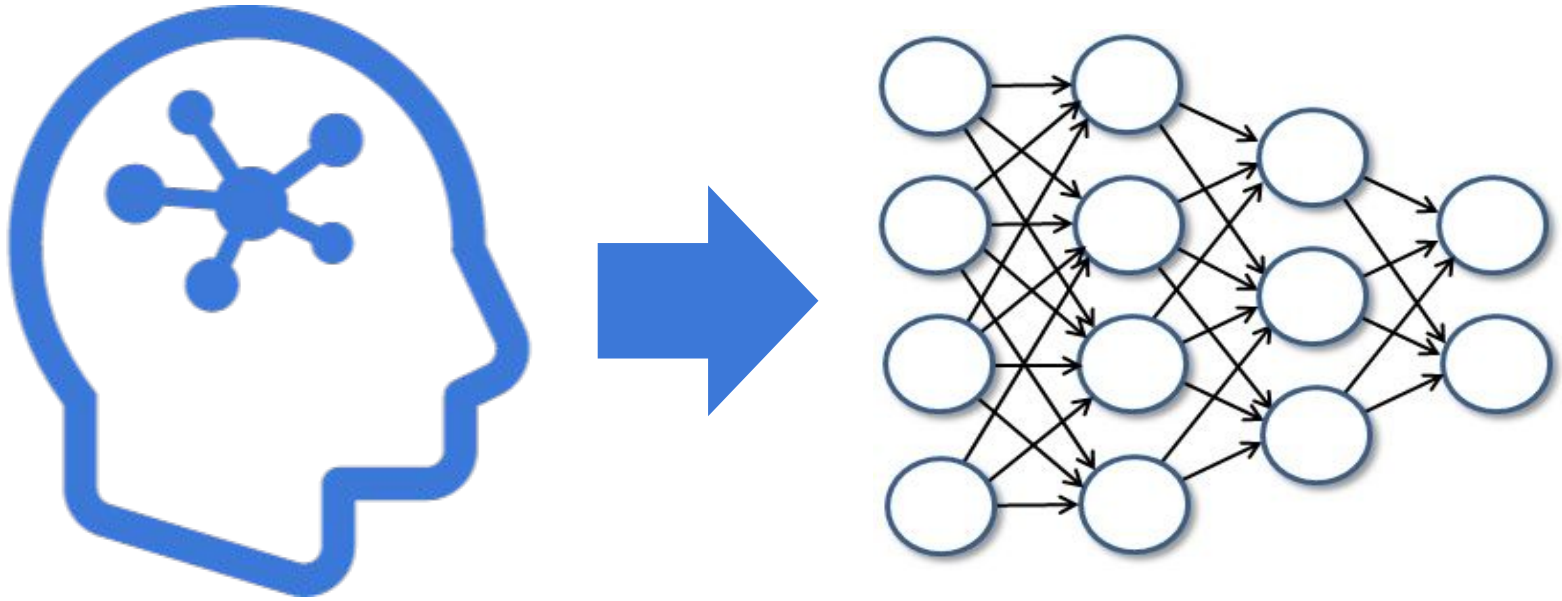
端的に言えば、**何かを識別（分類）する仕組み**



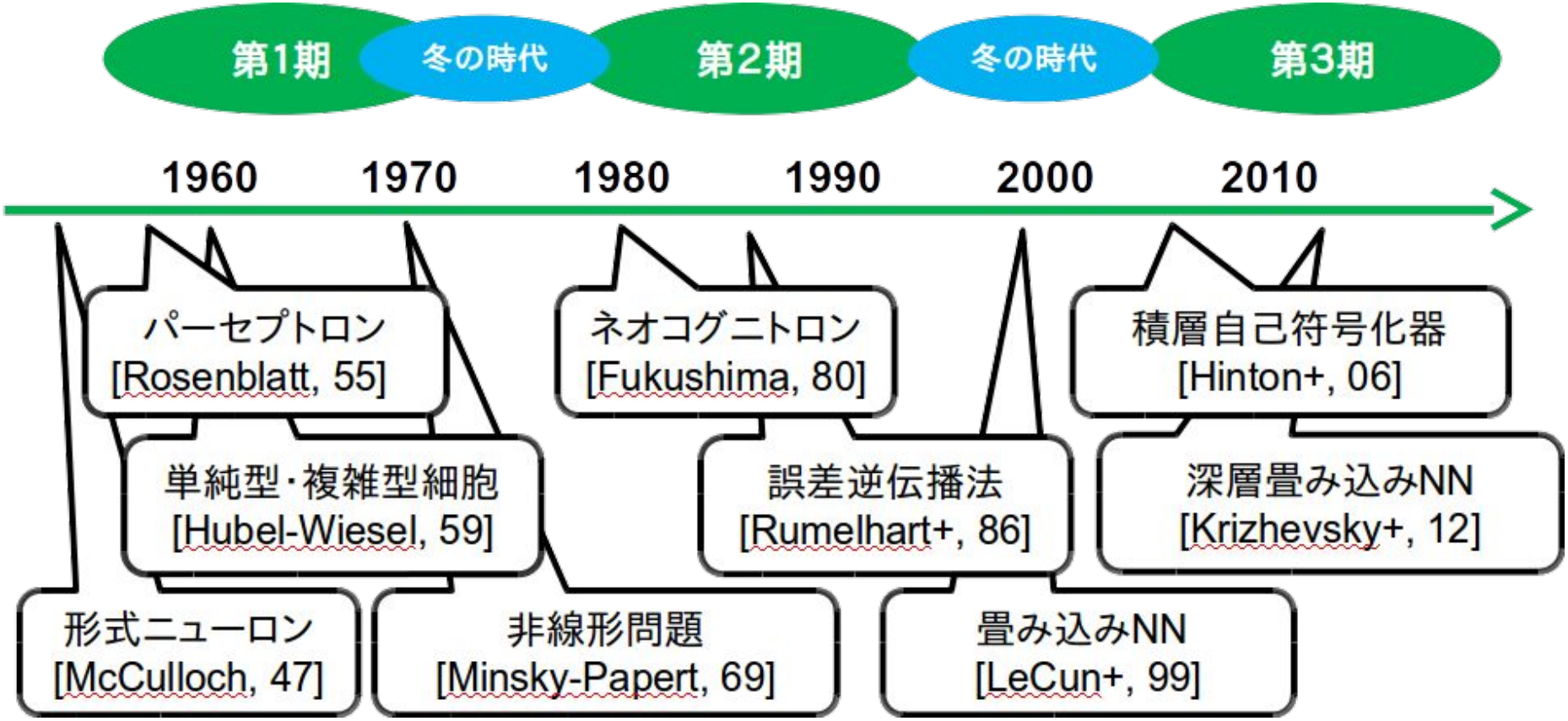
# ニューラルネットワーク（NN）

ニューラルネットワーク = 神経回路網

脳、目における情報処理の仕組みをコンピュータ上で再現する数学モデル



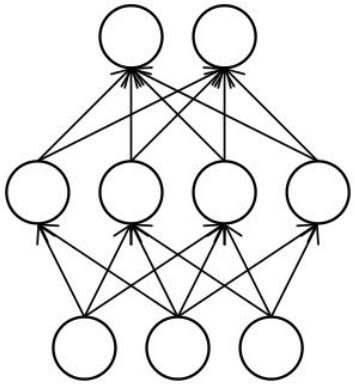
# NNの歴史


第1期
冬の時代
第2期
冬の時代
第3期
1960
1970
1980
1990
2000
2010
パーセプトロン
[Rosenblatt, 55]
単純型・複雑型細胞
[Hubel-Wiesel, 59]
形式ニューロン
[McCulloch, 47]
非線形問題
[Minsky-Papert, 69]
ネオコグニトロン
[Fukushima, 80]
誤差逆伝播法
[Rumelhart+, 86]
畳み込みNN
[LeCun+, 99]
積層自己符号化器
[Hinton+, 06]
深層畳み込みNN
[Krizhevsky+, 12]

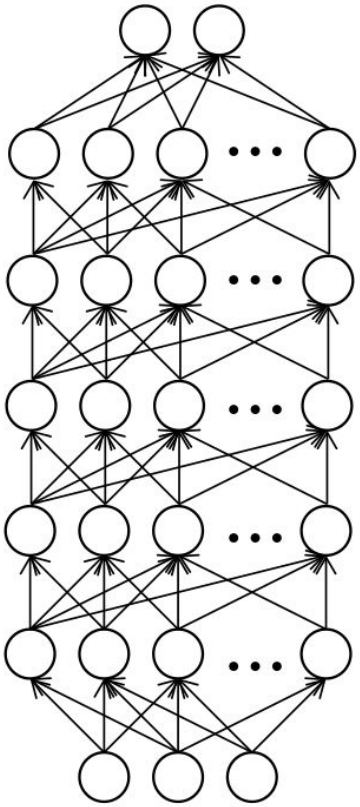
# Deep Learning

ニューラルネットワークの最先端技術

従来のネットワークをより **Deep（多層化）** にしているだけ？



浅いニューラルネット

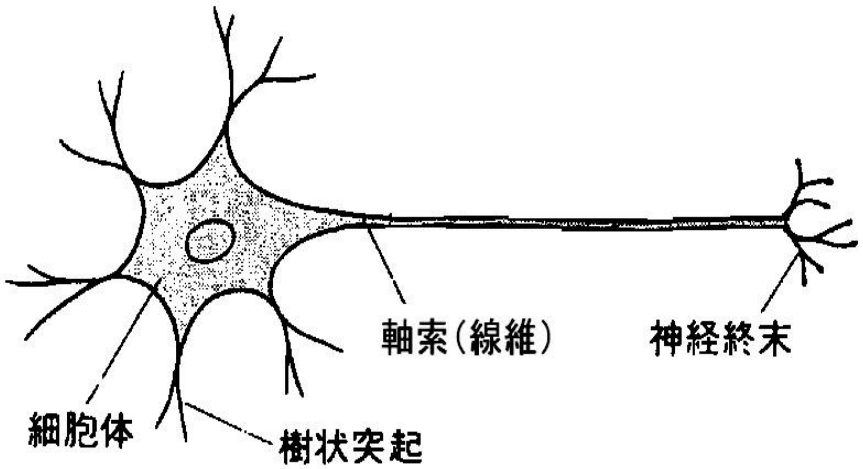


深いニューラルネット

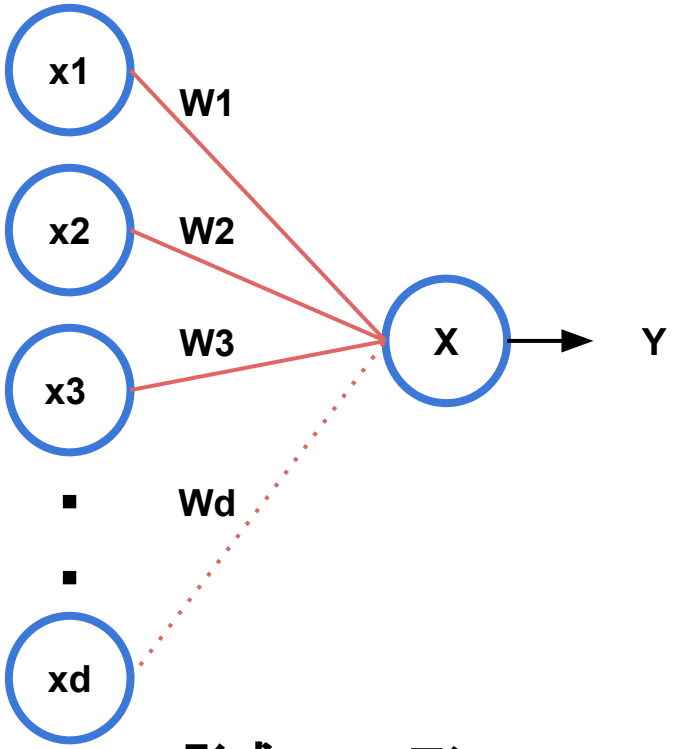
# 形式ニューロン

脳は無数のニューロン（神経細胞）で構成

McCullochらによってニューロンを数式化 [1943]



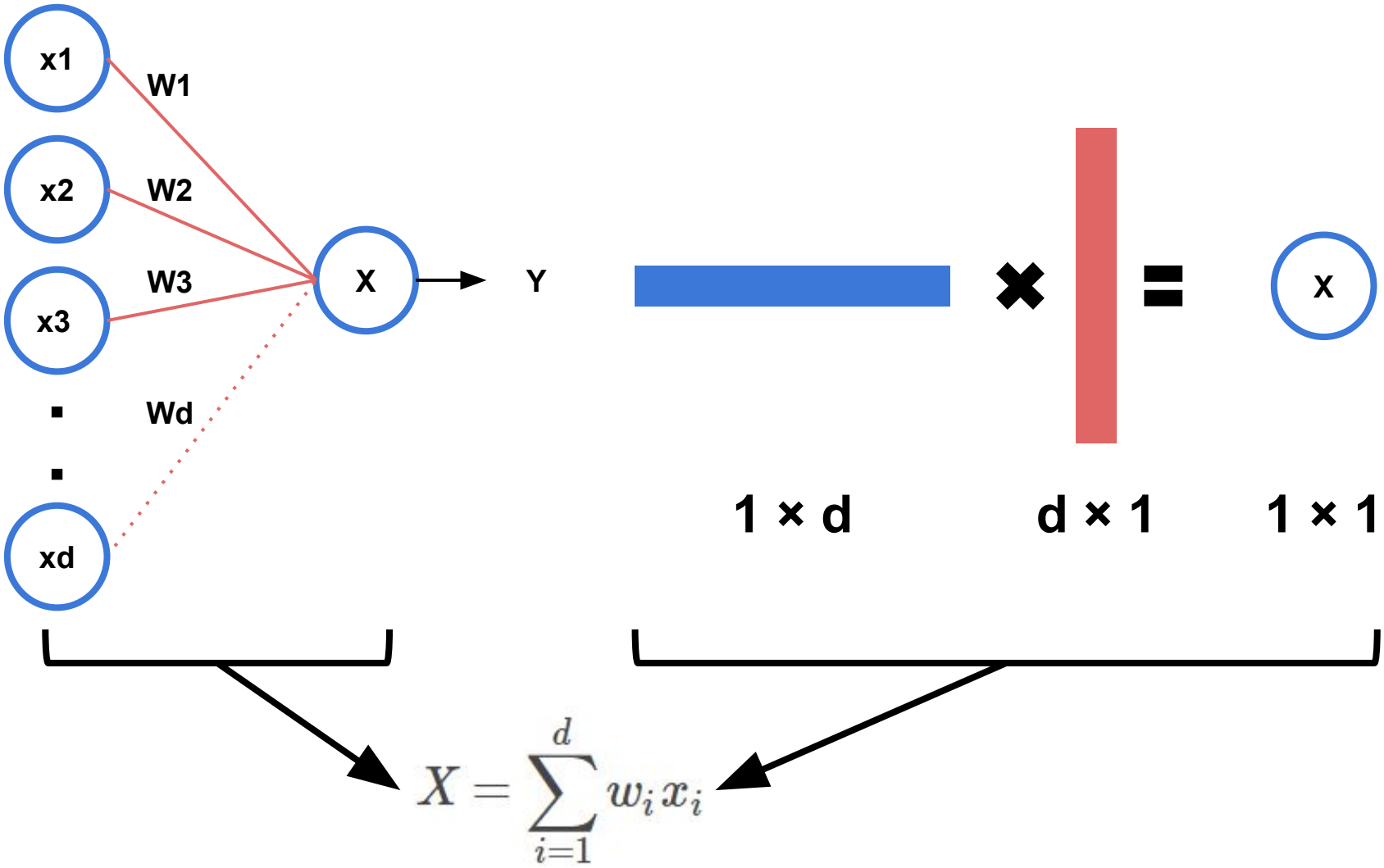

ニューロン




形式ニューロン

引用文献：http://www.keirinkan.com/kori/kori_biology/kori_biology_1_kaitei/contents/bi-1/3-bu/3-1-1.htm

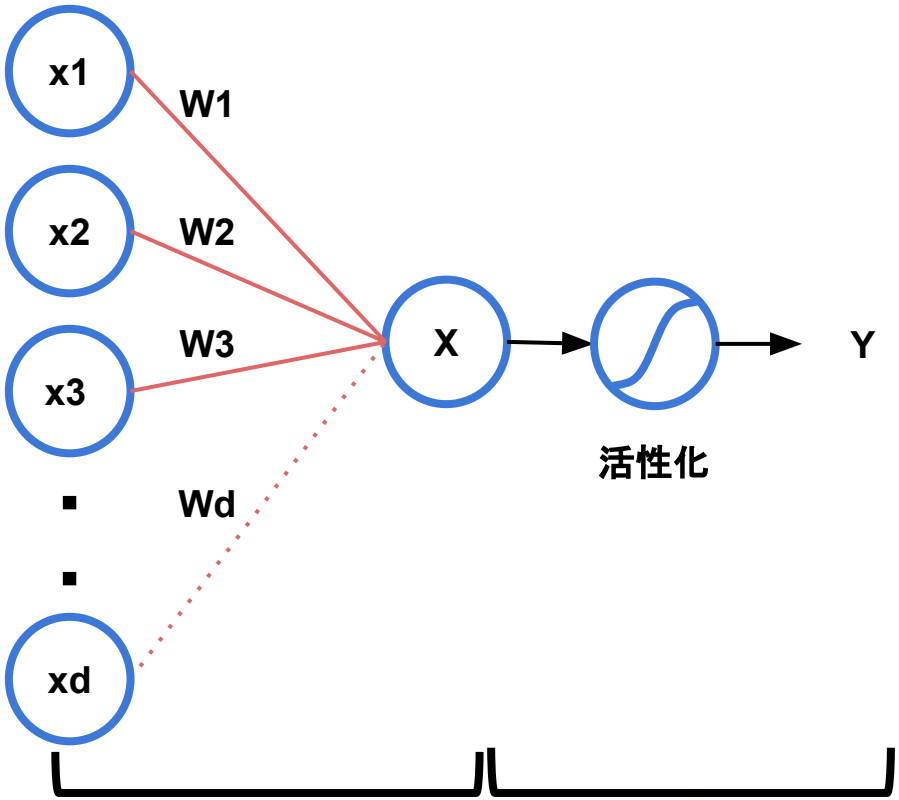
# 簡単な行列計算

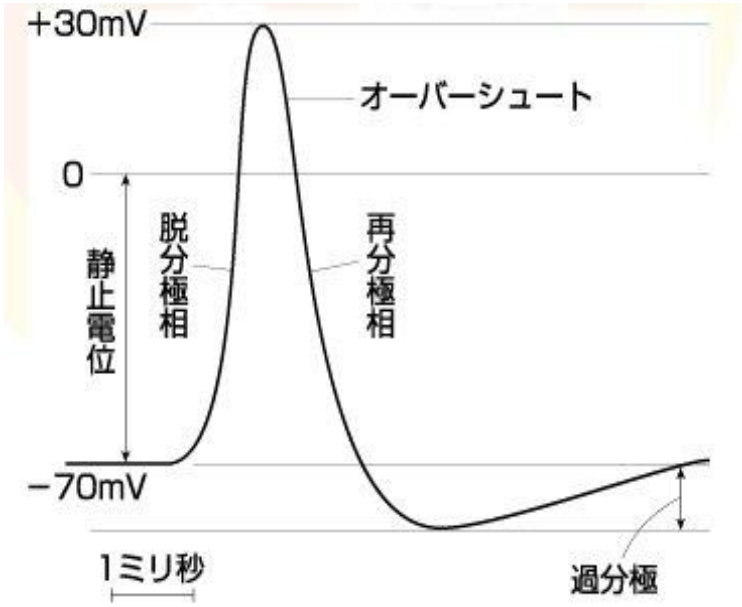



$$X = \sum_{i=1}^{d} w_i x_i$$

# 活性化関数(1)

ニューロンは活性化(スパイク)で信号伝達



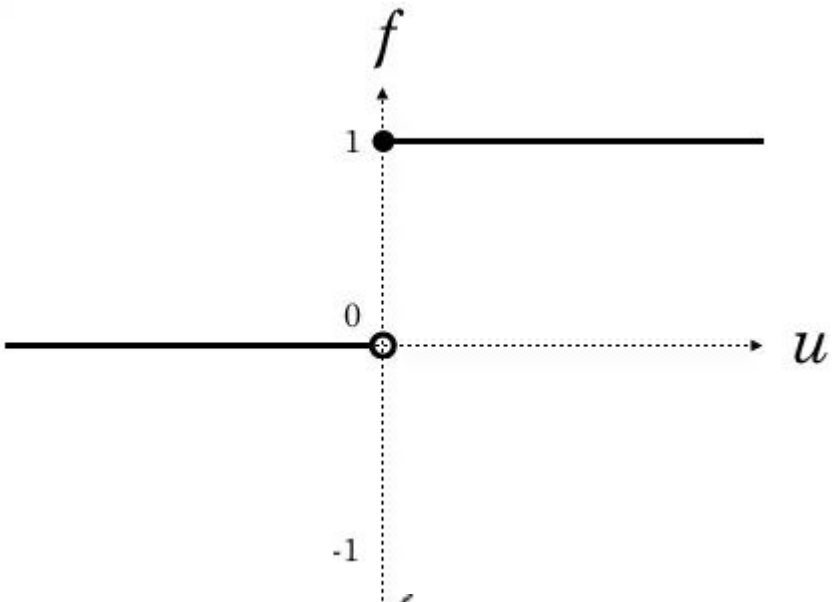

▲ 活動電位の経過

$$X = \sum_{i=1}^{d} w_i x_i \qquad y = f(\sum_{i=1}^{d} w_i x_i)$$

# 活性化関数(2)

ステップ関数

- 0か1の出力　→ 閾値を超えた場合にのみ発火する
- 単純パーセプトロンの活性化関数はこれ

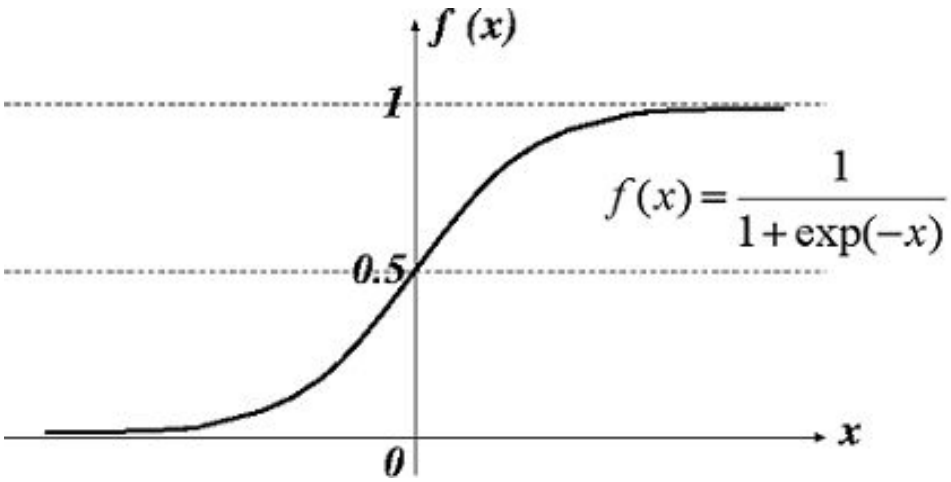


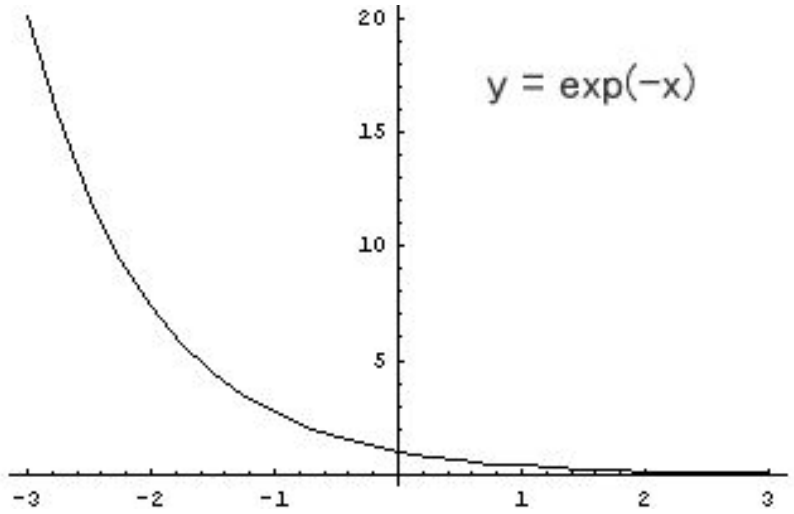

$$f(u) = \begin{cases} 1 & (u \geq 0) \\ 0 & (u < 0) \end{cases}$$

# 活性化関数(2)

シグモイド関数によって 0~1 の値に変換

ステップ関数の出力は0か1であることと比較して、　　元の入力の値を殺しすぎない

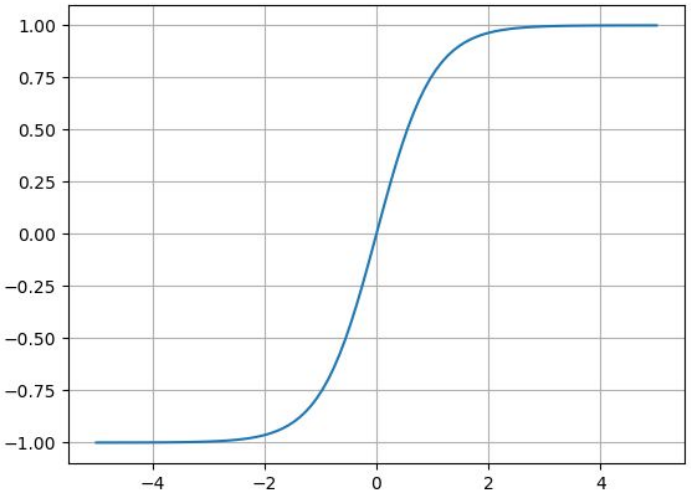


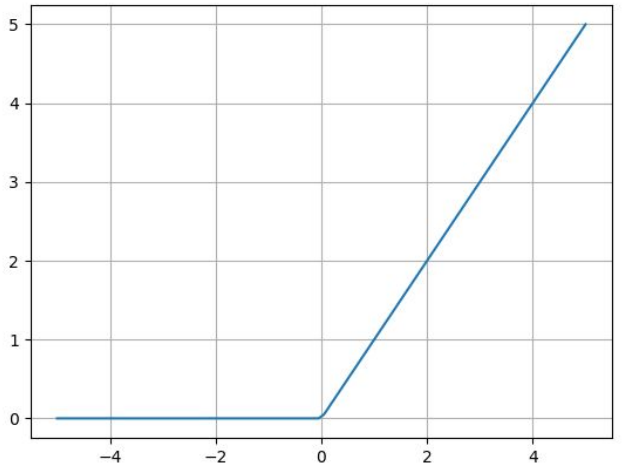

$$f(X)=\frac{1}{1+\exp(-gX)}$$

# 活性化関数(3)

tanh関数



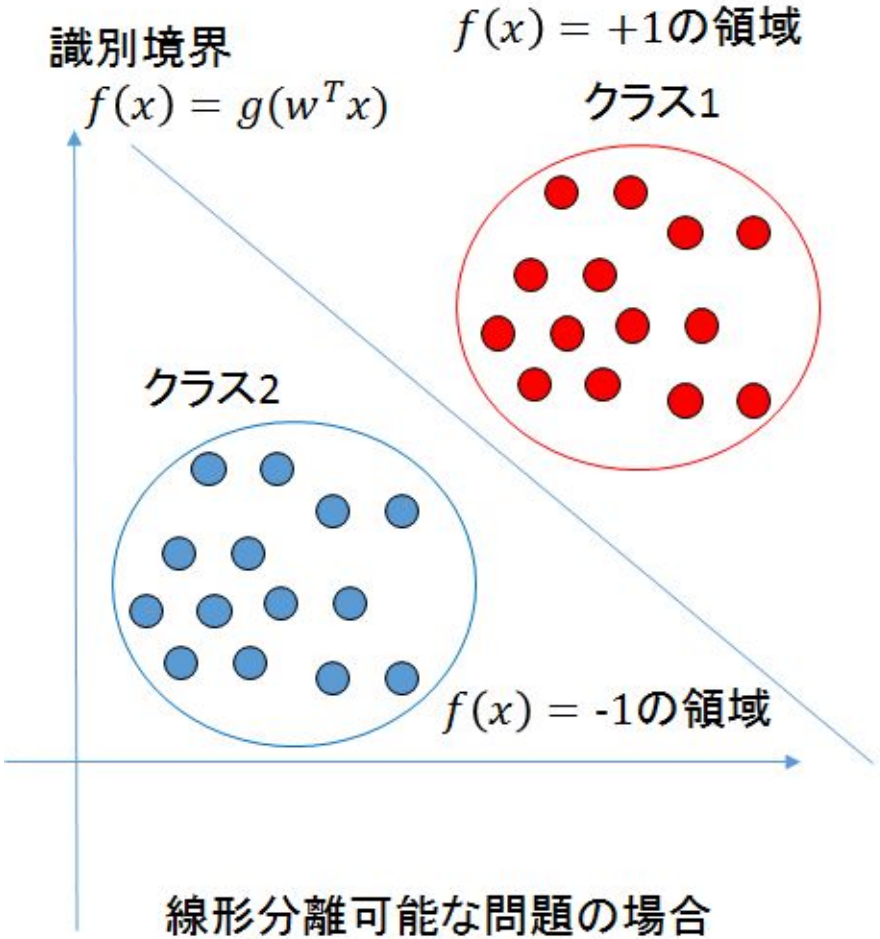

ReLU関数



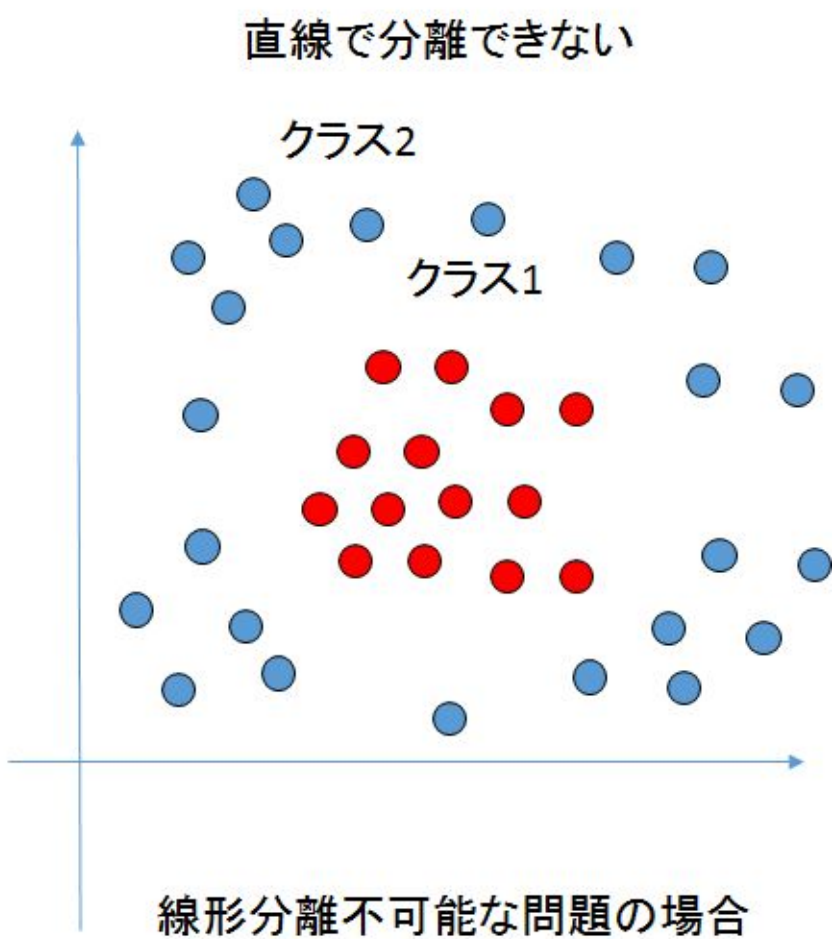

シグモイド関数以降の活性化関数に総じて言えることは

**微分可能な関数**であるということ(後に重要となる要素)

# パーセプトロンの限界

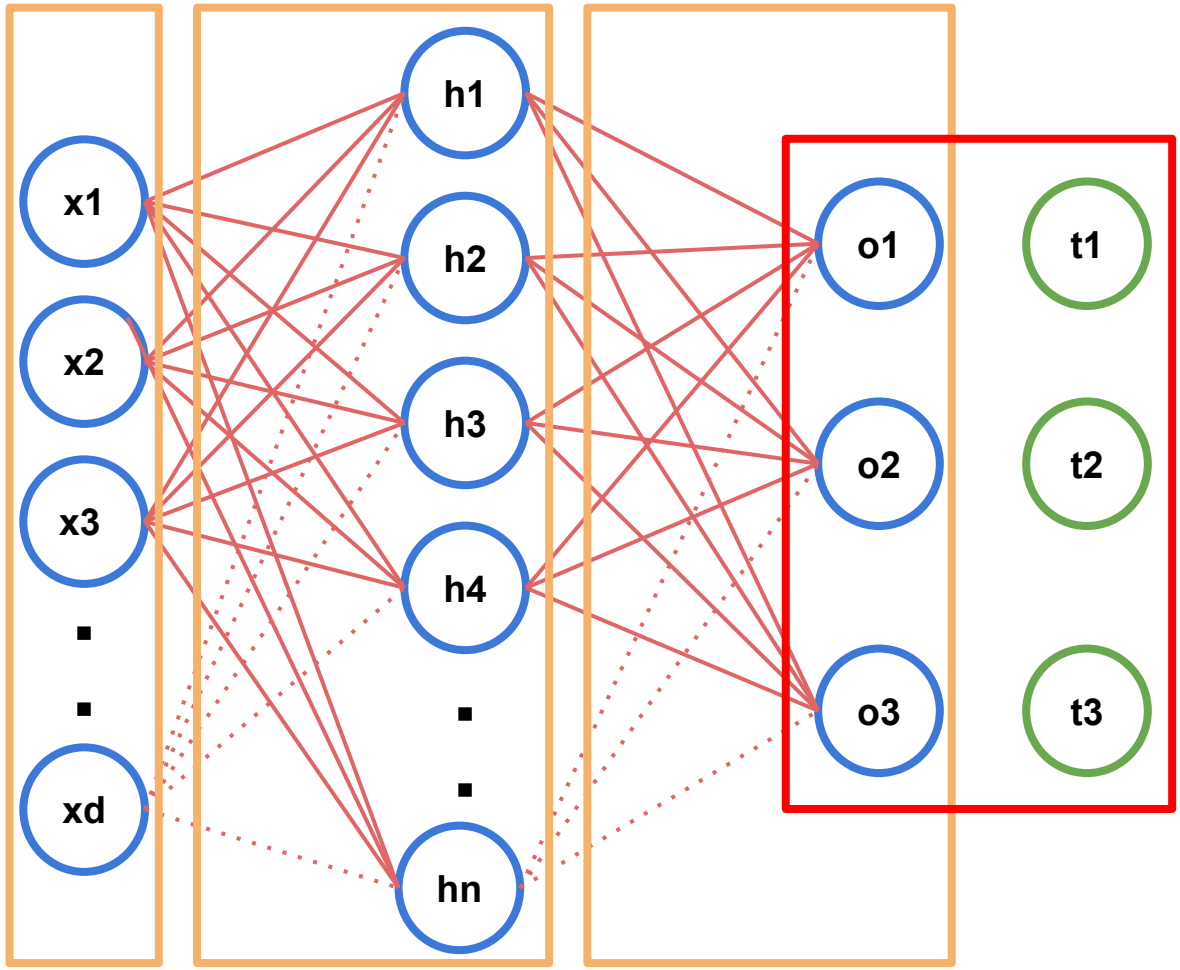



線形分離可能な問題の場合



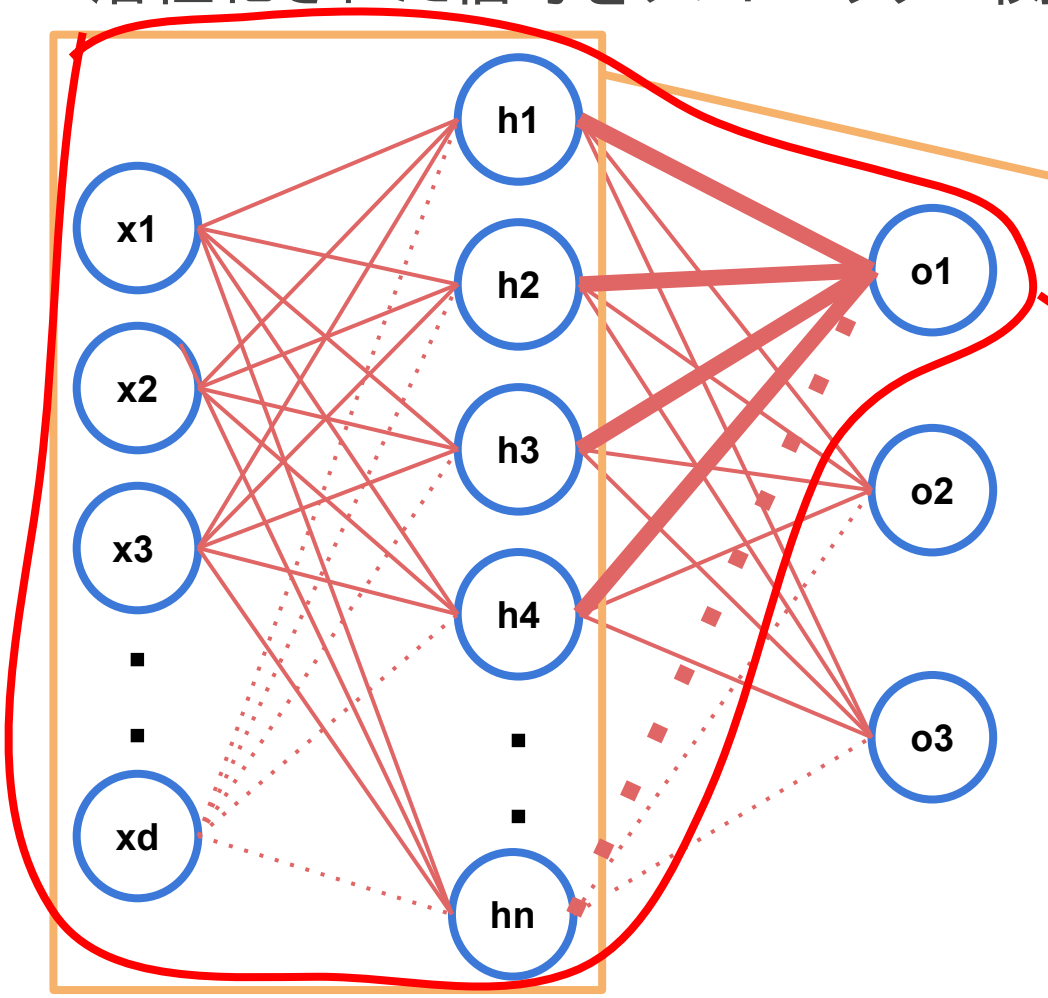

線形分離不可能な問題の場合

# 多層パーセプトロン


x1
x2
x3
xd
h1
h2
h3
h4
hn
o1
o2
o3
t1
t2
t3
入力層
中間層
（隠れ層）
出力層
（ソフトマックス関数）
教師信号

# ソフトマックス関数

活性化された信号をソフトマックス関数で確率化




$$A_j = f(\sum_{i=1}^{d} w_{ij} S_i + \theta_j)$$

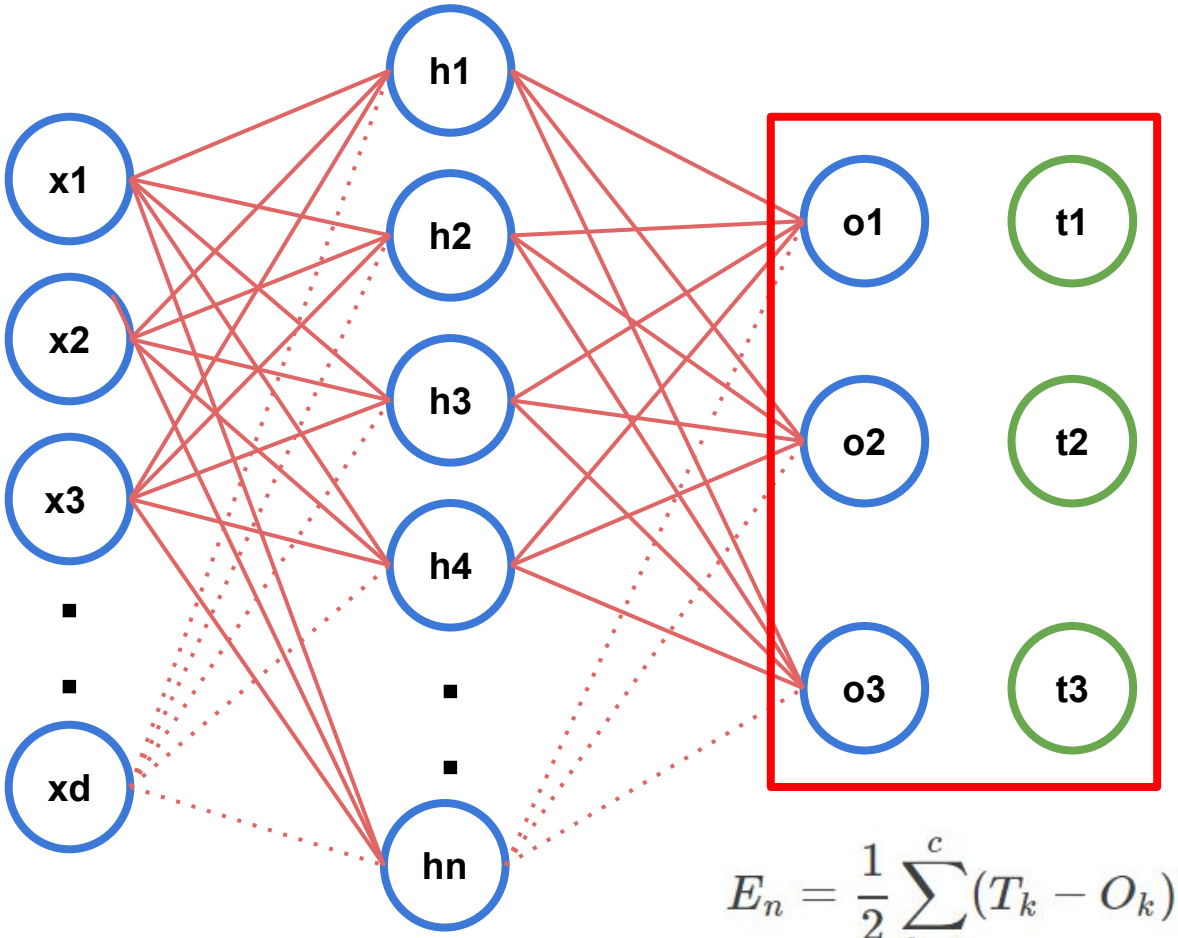
$$P_k = \sum_j w_{jk} A_j + \gamma_k$$

$$O_k = \frac{\exp(P_k)}{\sum_j \exp(P_j)}$$

**ソフトマックス関数**

# 誤差関数

出力と教師信号との差を求めてネットワークを修正していく




$$E_n = \frac{1}{2}\sum_{k=1}^{c}(T_k - O_k)^2$$

# 勾配降下法

誤差関数を微分して目的パラメータの勾配を計算

学習速度と勾配の乗算を目的パラメータから引くことで調整

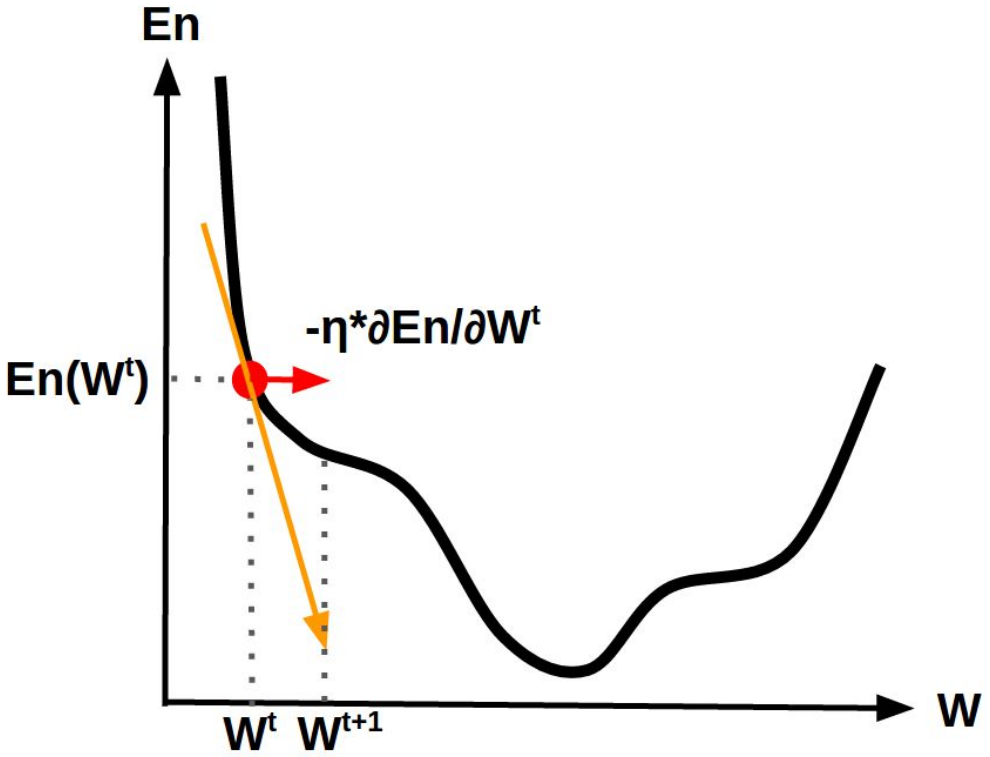
$$E_n = \frac{1}{2}\sum_{k=1}^{c}(T_k - O_k)^2$$

**誤差関数**

$$w^{t+1} = w^t - \eta \frac{\partial E_n}{\partial w^t}$$

**更新式**

**微分を前提としている！！**




引用文献：http://www.neuro.sfc.keio.ac.jp/~masato/study/SVM/grad.htm

# Mnist

Mnist（手書き文字）認識をMLPで解く

機械学習関連のアルゴリズムのベンチマークデータ

何かモデルを構築した際は、このデータで動作確認すると良い

# 実装環境

OSはUbuntuを使用 [untitled:Untitled-1]

・環境構築が楽（pythonは標準で搭載）

・OpenCV等ライブラリもコマンドでインストール可

例： [ sudo pip install python-opencv] でおk

自分はUbuntuマシン上でWindowsの仮想環境を構築してるヨ

Pythonで実装

コーディング数が少なくて開発効率が高い

**Numpy**、Scipyなどの主要ライブラリはバックエンドでC++が動いている

GPU計算が必須となるため、計算速度は気にするな

# DL実装にあたって使用するライブラリ

ライブラリを使用することでコーディングを短縮化

Caffe：BVLC開発（Python、C++、Matlab）

古参のライブラリ、研究コミュニティが広い

Theano：モントリオール大学開発（Python）

数値計算ライブラリ、自動微分を実現

**Keras ( Backend :Tensorflow ) Google開発（Python、C++）**

モデルの可視化、GPUの対応、商用利用が楽

Chainer：Preferred Networks開発（Python）

コーディングがシンプル、インストールが楽

# パーセプトロンを実装しよう

pythonに慣れるためにコードを書いていきます

# 次回予定

Python環境構築・使い方

　　list Numpy

データ整形

CIFAR10 Caltech101

CNN

今日の資料は後日nasに投下します

# 課題

実装したパーセプトロンがどのような関数になっているか プロットして確認するPGを作成

np.arange()

plt.plot(Y)

# 参考

http://www.vision.is.tohoku.ac.jp/files/9313/6601/7876/CVIM_tutorial_deep_learning.pdf