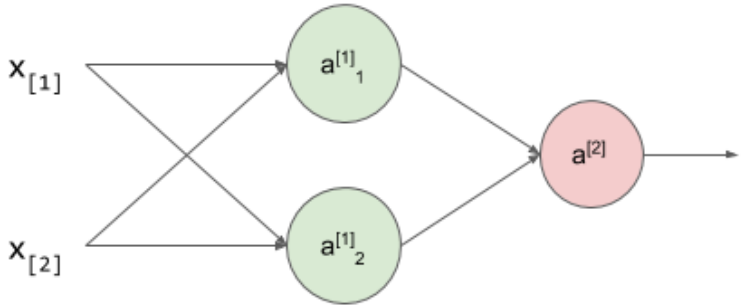
深層学習入門　～第一回～

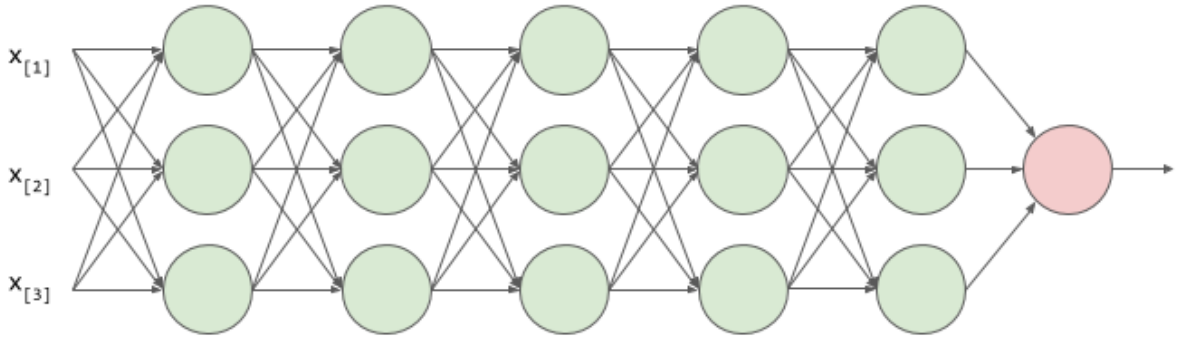
東海大学情報通信学研究科情報通信学専攻

島田　建

深層学習とは

- 強力な関数近似器
- 人間の脳の構造を模した機械学習手法の一つである





Neural network templates

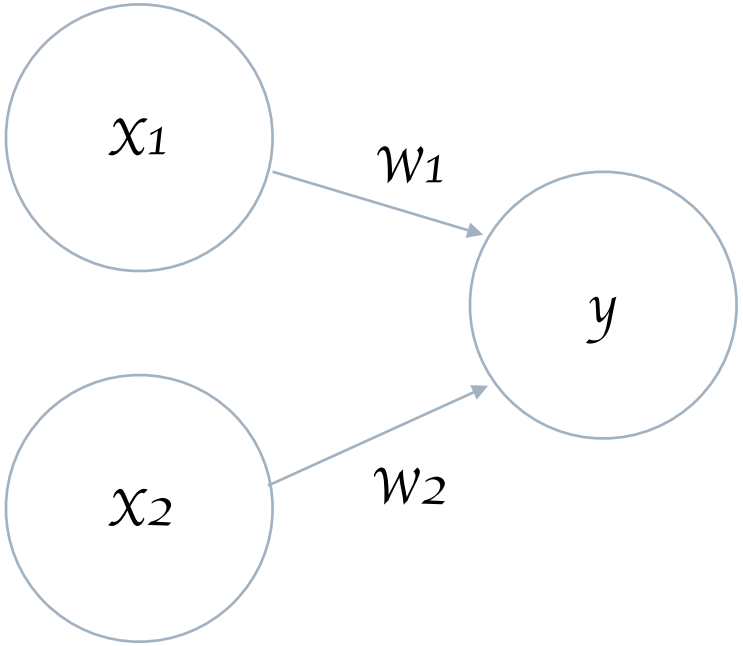
パーセプトロンとは

- 複数の信号を入力として受け取り、一つの信号を出力する
- 「信号」は電流や川の「流れ」をイメージ
- パーセプトロンは「流す/流さない（1/0）」の値しか持たない
- 閾値を超えた際に 1 を出力する

x_1, x_2 : 入力信号

w_1, w_2 : 重み

y : 出力信号



$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leqq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

パーセプトロンとは

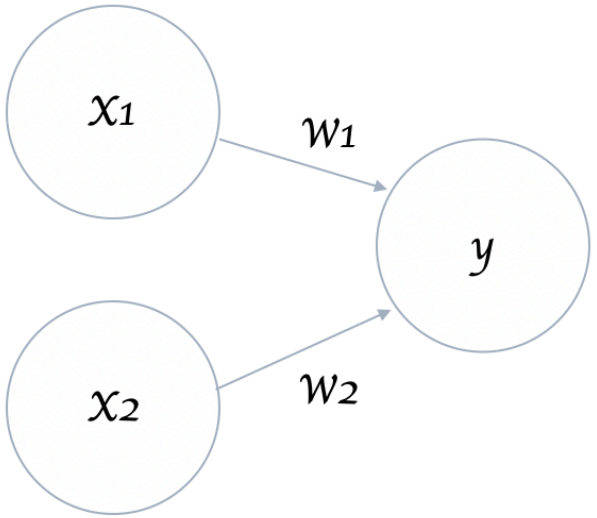
- ANDゲートをパーセプトロンで表現
- ORやNANDも同様に表現可能

ANDゲートの真理値表

x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1

$(w_1, w_2, \theta) = (0.5, 0.5, 0.7)$や

$(w_1, w_2, \theta) = (0.5, 0.5, 0.8)$などで表現できる



パーセプトロンの実装

- Numpyを用いて簡単に実装可能
- 先の式をベースに実装
- あとで実行してみて真理値表と同じものが得られることを確認しよう

```python
# ANDゲート
def AND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = -0.7
    tmp = np.sum(w * x) + b
    if tmp <= 0:
        return 0
    else:
        return 1
```

ANDゲートの真理値表

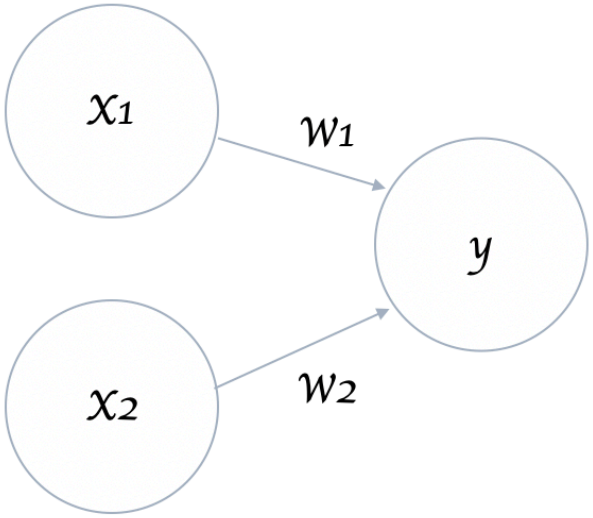
x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1

パーセプトロンでXORを表現

- XORゲートをパーセプトロンで表現してみよう
- 真理値表は図のようになっている

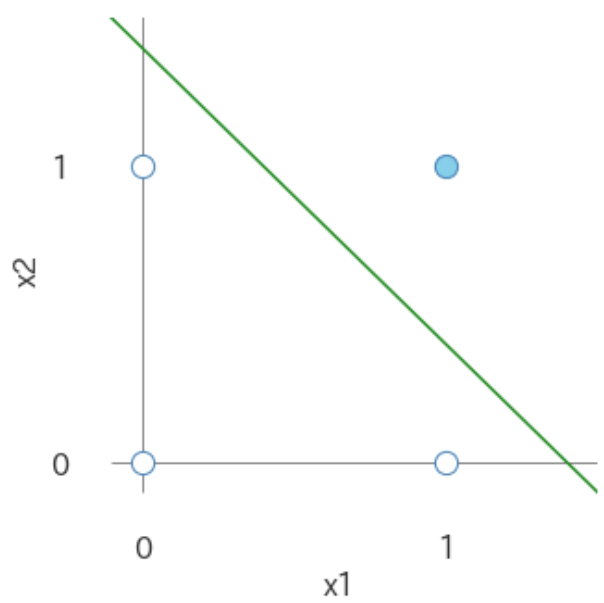
XORゲートの真理値表

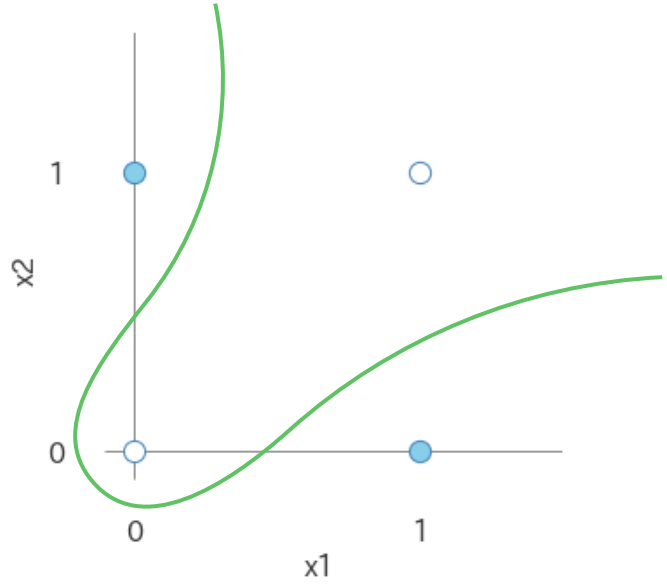
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0



線形・非線形

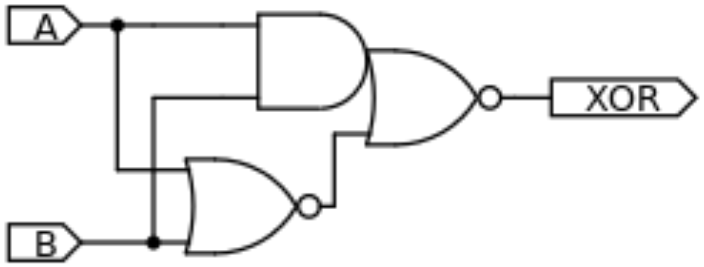
- XORは単一のパーセプトロンでは表現することができない
- 左図はANDをプロットしたもので、右図はXORをプロットしたものである
- ANDは直線1本で領域を分離できるが、XORは直線1本では分離できない
- 直線1本で領域を分離できるものを”線形”といい、分離できないものを”非線形”という
- 今回の場合なら、ANDは”線形”な領域であり、XORは”非線形”な領域である

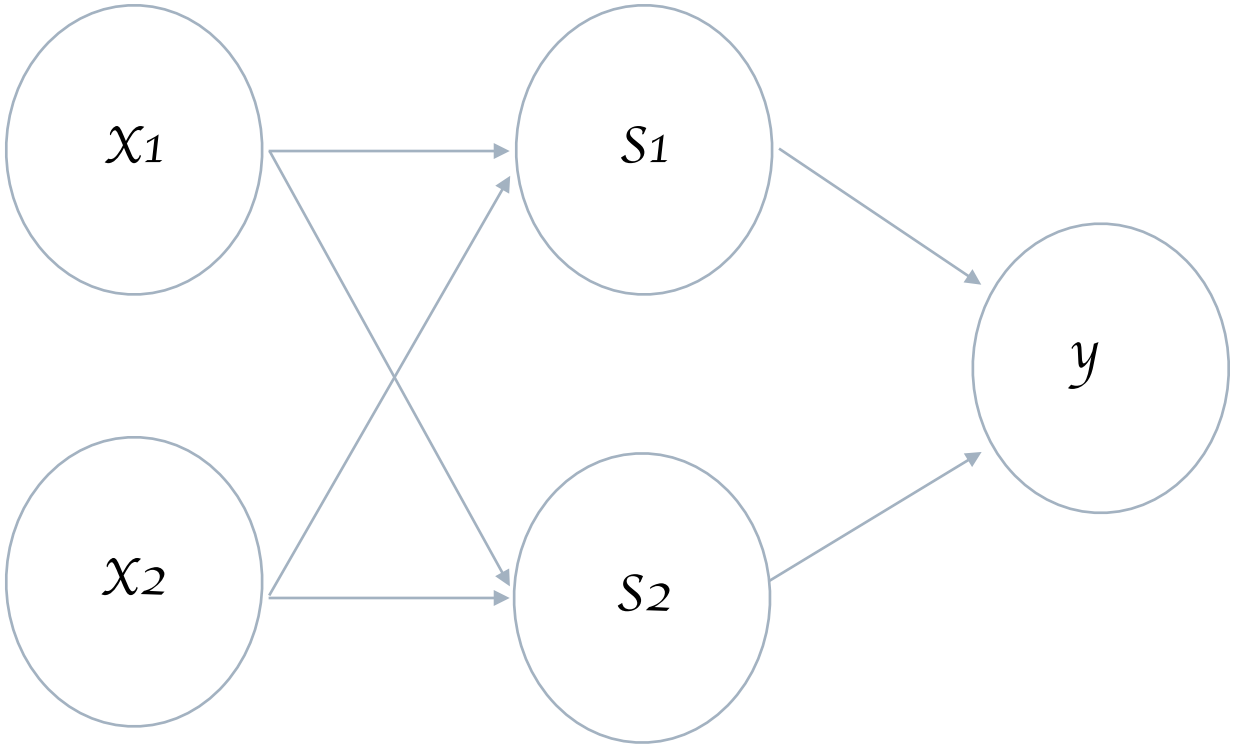




XORゲートの実装

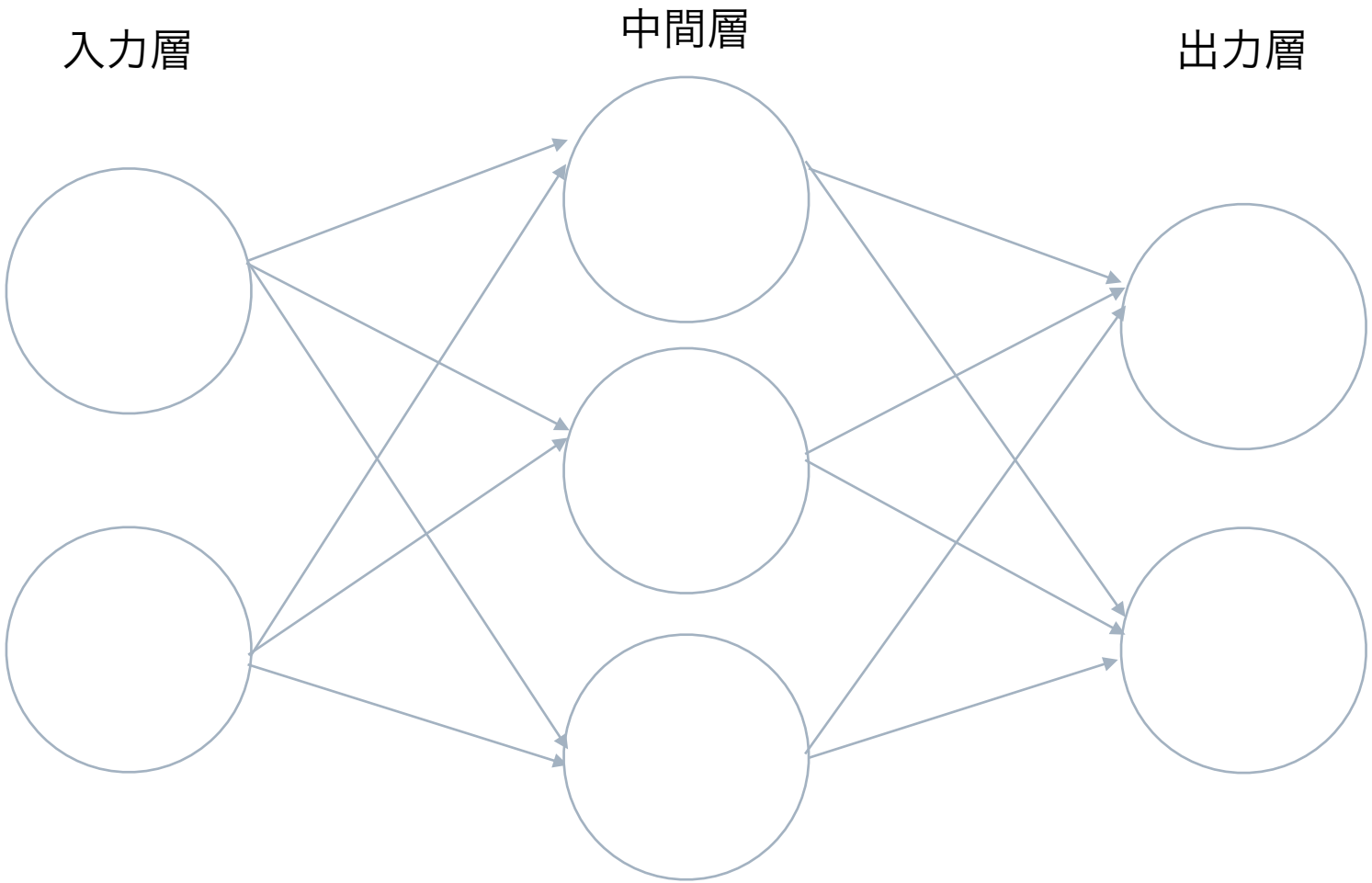
- AND、NAND、ORを用いることで実装可能
- 単層のパーセプトロンを重ね、多層のパーセプトロンとすることでXORを表現可能





ニューラルネット

- 先のパーセプトロンでは重みを人手で設定する必要があった
- ニューラルネットでは重みを自動で学習することができる



パーセプトロンの復習

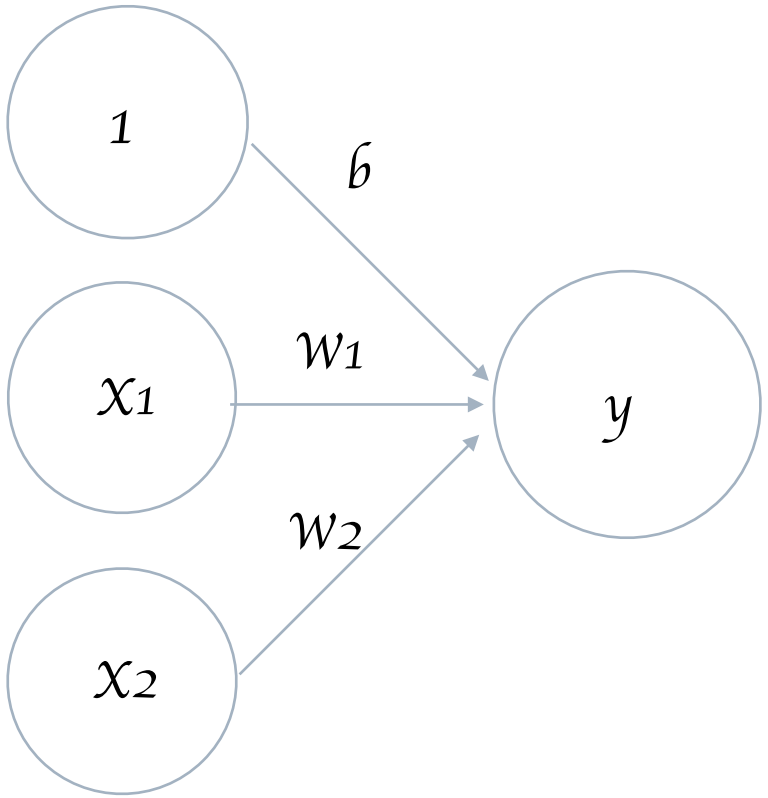
- bはバイアスと呼ぶ
- バイアスは出力信号が１を出力する度合いを調整するパラメータ

x_1, x_2 : 入力信号

w_1, w_2 : 重み

y : 出力信号

b : バイアス



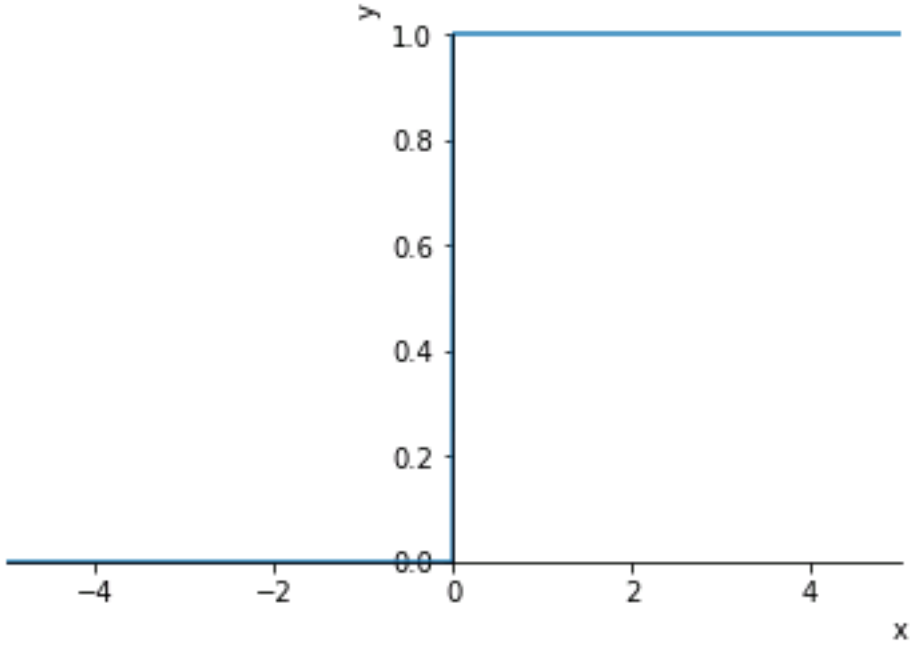
$$y = \begin{cases} 0 & (b + w_1x_1 + w_2x_2 \leqq 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases}$$

活性化関数

- 入力信号の総和を出力信号に変換する関数のこと
- パーセプトロンでは活性化関数にステップ関数を用いている
- 活性化関数では非線形活性化関数を用いる

$$y = h(b + w_1 x_1 + w_2 x_2)$$

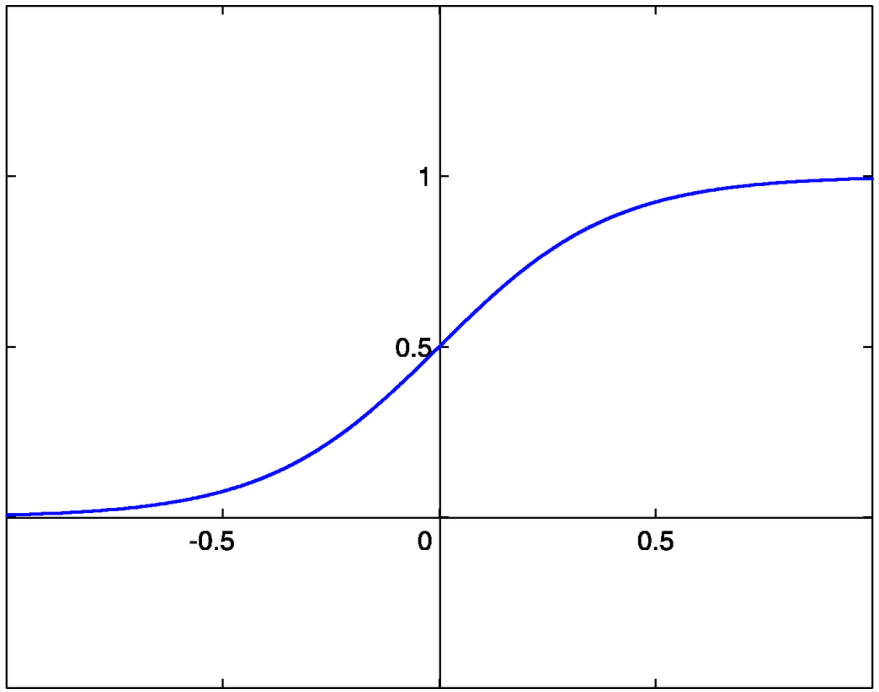
$$h(x) = \begin{cases} 0 & (x \leqq 0) \\ 1 & (x > 0) \end{cases}$$



シグモイド関数

- 活性化関数の一つ
- exp(-x)はe^{-x}のこと

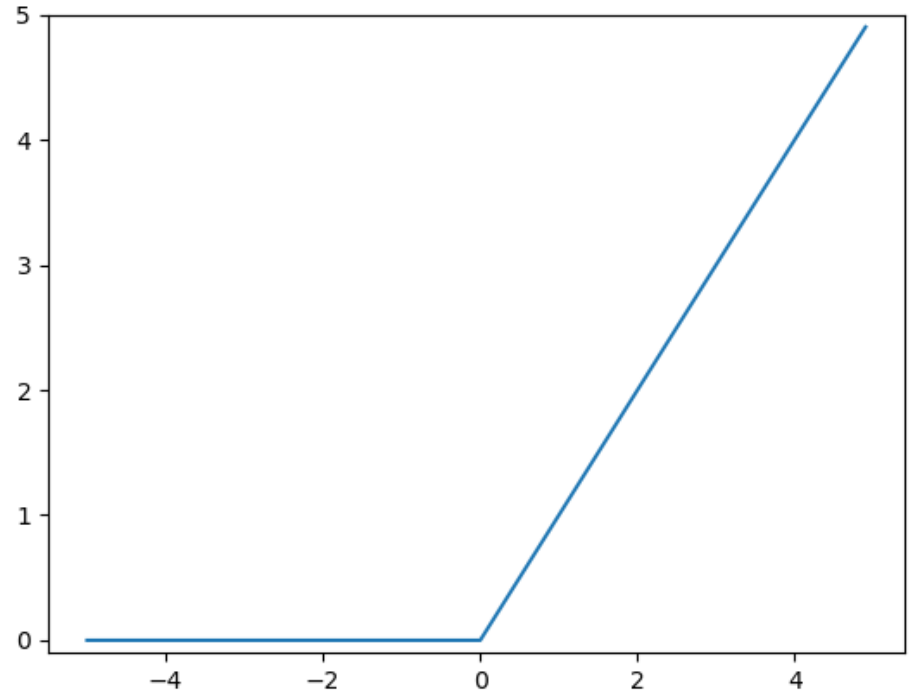
$$h(x) = \frac{1}{1 + exp(-x)}$$



ReLU関数

- ReLUは入力が0より上ならば入力をそのまま出力し、0以下ならば0を出力する関数

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



softmax関数

- Softmax関数では出力が0.0~1.0の範囲に収まる
- この関数の出力の総和は1.0になるため「確率」として考える場合が多い

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

活性化関数の線形・非線形について

- 線形な活性化関数をいくら重ねても意味がない
- 層を重ねることの意味を出すために非線形な活性化関数を用いる必要がある

$$h(x) = ax$$

活性化関数h(x)を3層重ねる

$$y(x) = h(h(h(x)))$$

中身を計算

$$y(x) = a \times a \times a \times x$$

同じことを1層で表現可能

$$y(x) = bx \, (b = a^3)$$

損失関数

- ネットワークの性能の”悪さ”を示す指標
- この損失を最小化することがニューラルネットの学習
- 任意の関数を用いることができるが、一般的には後述する２つの関数が用いられる

Mean Squared Error

- 日本語では２乗和誤差
- y_kはネットワークの出力、t_kは正解ラベル、kはデータの次元数

$$E = \frac{1}{2}\sum_k (y_k - t_k)^2$$

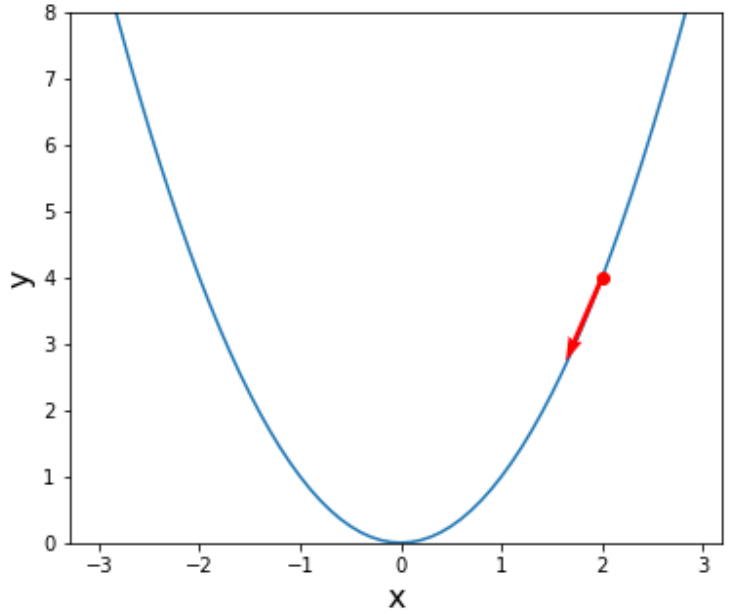
Cross Entropy Loss

- 日本語では交差エントロピー誤差
- Logは底がeの自然対数を表す

$$E = -\sum_{k} t_k \log y_k$$

勾配降下法

- 勾配が示す方向は、関数の値をもっとも減らす方向
- 勾配法とは勾配の方向に一定距離だけ進む
- 最小値を探すのか最大値を探すのかで名称が異なるが深層学習の分野では勾配降下法と呼ばれることが多い



$$x_0 = x_0 - \eta \frac{\partial f}{\partial x_0}$$

$$x_1 = x_1 - \eta \frac{\partial f}{\partial x_1}$$

最適化

- ニューラルネットの学習は損失関数の値をできるだけ小さくすること
- これは最適なパラメータを見つけることであり、この問題を解くことを”最適化”という
- 先の勾配をもとにパラメータを更新していく
- 最適化の手法はいくつか存在している

SGD

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

AdaGrad

$$h \leftarrow h + \frac{\partial L}{\partial W} \odot \frac{\partial L}{\partial W}$$

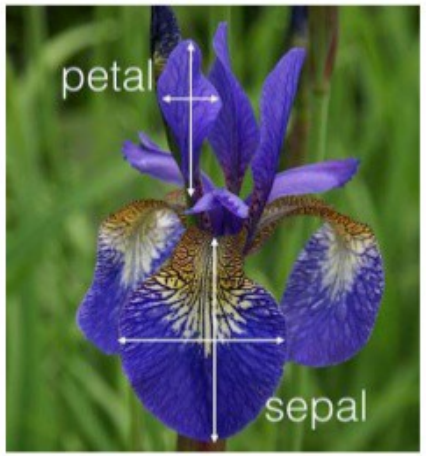
$$W \leftarrow W - \eta \frac{1}{\sqrt{h}} \frac{\partial L}{\partial W}$$

実習

- Irisdatasetを用いて簡単なニューラルネットを実装してみる
- 本講義ではPyTorchを用いて実装を行う

https://pytorch.org

Supervised learning *classification* problem
(using the Iris flower data set)



Training / test data

Features				Labels
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	Iris setosa
4.9	3.0	1.4	0.2	Iris setosa
7.0	3.2	4.7	1.4	Iris versicolor
6.4	3.2	4.5	1.5	Iris versicolor
6.3	3.3	6.0	2.5	Iris virginica
5.8	3.3	6.0	2.5	Iris virginica

IrisDataset