深層学習基礎のあんちょこ

DEEP-PEOPLE #4

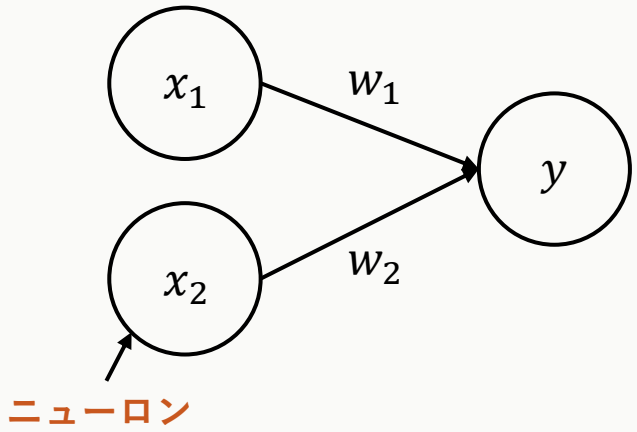
2022/4/27

深層学習を始める前に

パーセプトロン

パーセプトロン(Perceptron)

- **複数の信号を入力として受け取り，1つの信号を出力する**
  - 2つの入力の重み付き和がある値(閾値)を越えるかで0/1を出力する
  - $y = 1$のとき「ニューロンが発火した」という

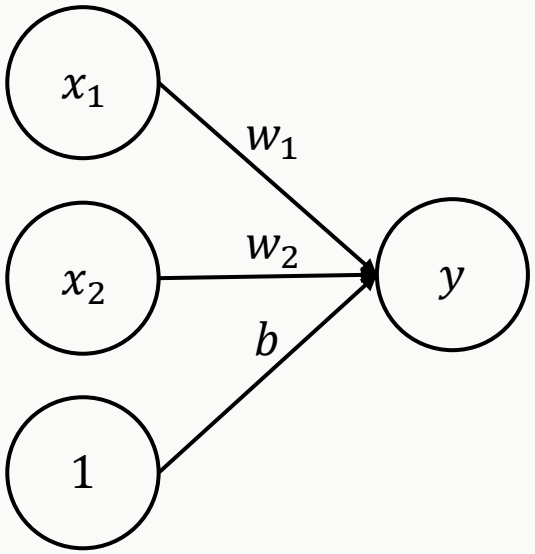


$$y = \begin{cases} 0\ (w_1x_1 + w_2x_2 \leq \varphi) \\ 1\ (w_1x_1 + w_2x_2 > \varphi) \end{cases}$$

y ：出力
x_1, x_2 ：入力
w_1, w_2 ：重み
φ ：閾値

パーセプトロン ver.2

- 前ページの閾値φを$-b$に変えると良い感じにわかりやすくなる
  - bは**バイアス(Bias)**と呼ばれ，ニューロンの発火しやすさを表す



$$y = \begin{cases} 0 \ (b + w_1 x_1 + w_2 x_2 \leq 0) \\ 1 \ (b + w_1 x_1 + w_2 x_2 > 0) \end{cases}$$

深層学習を始める前に

パーセプトロンの使い道

- **単層パーセプトロンで論理回路を表現できる！**
  - 重みを適切に設定することでAND, NAND, ORゲートを表現可
  - (NANDゲートの組み合わせですべての論理回路を表すことができる)
    - 究極的にはNANDゲートだけでコンピュータを作り出せる(NAND2Tetris, ...)

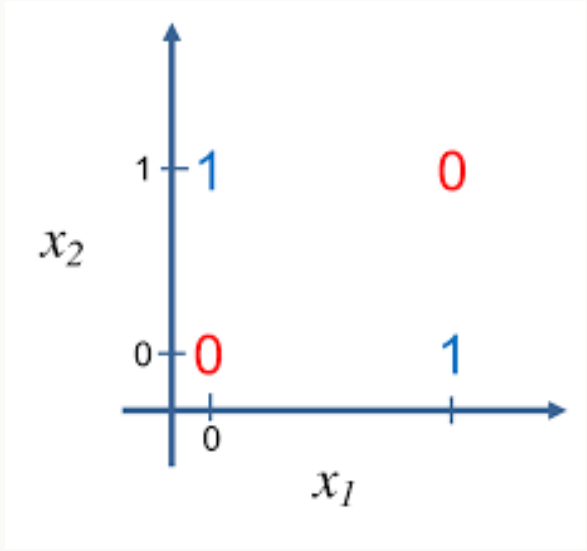
→XORを表現してみよう

単層パーセプトロンでXORを表す

- 単層パーセプトロンで表現できる=出力を線形関数で分けられる

▼XORゲートの真理値表[1]

INPUT 1	INPUT 2	OUTPUT
1	1	0
1	0	1
0	1	1
0	0	0



▲入力に対するXORゲートの出力[1]

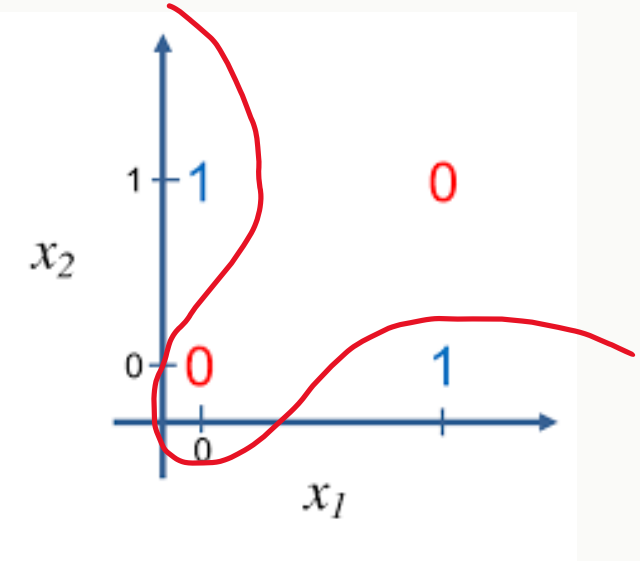
分けられなくない？

多層パーセプトロン

- 単層パーセプトロンでは XORゲートを表現できない
  - AND, NAND, ORゲートを組み合わせることでXORゲートを表現できる

  ⇔ 単層パーセプトロンを積み重ねる　〃

多層パーセプトロン：非線形関数への拡張
右図の赤線を表現できる



活性化関数

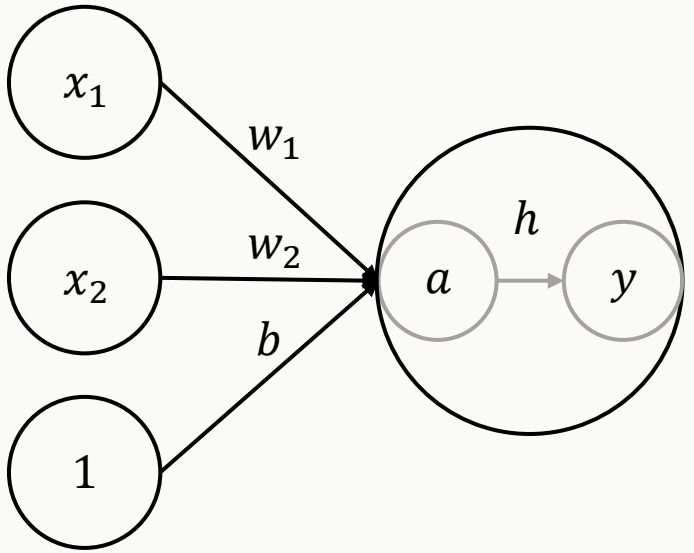
- 単層パーセプトロンの式を以下のように書き換える

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

- 非線形関数hを**活性化関数**と呼ぶ

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

  ならば単層パーセプトロン(ステップ関数)

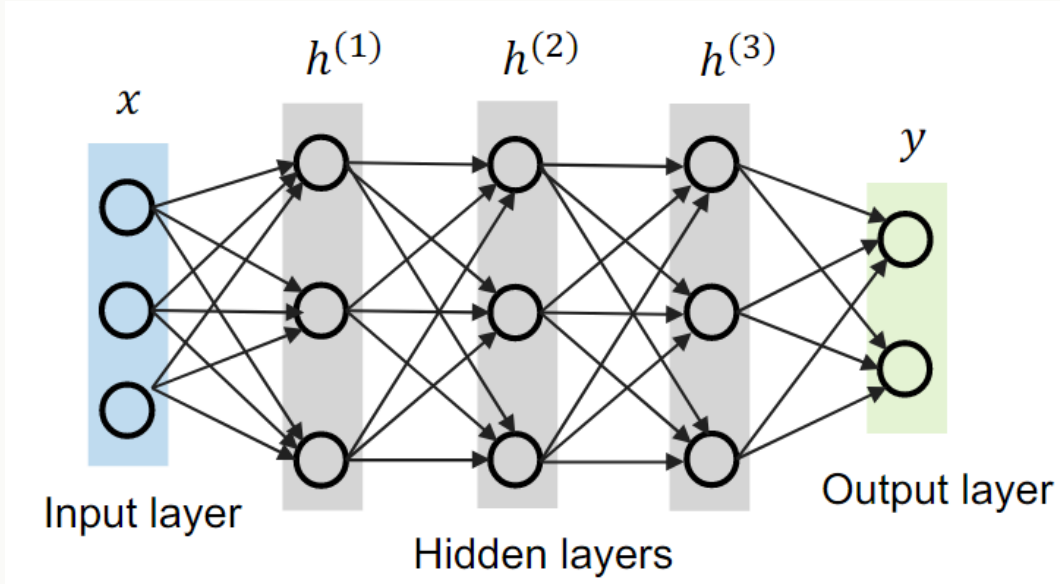


深層学習を始める前に

ニューラルネットワーク

ニューラルネットワーク(Neural Network, NN)

- NNは全結合層(パーセプトロン)と活性化関数で構成される
  - Feedforward Neural Network, Deep Feedforward Networkとも



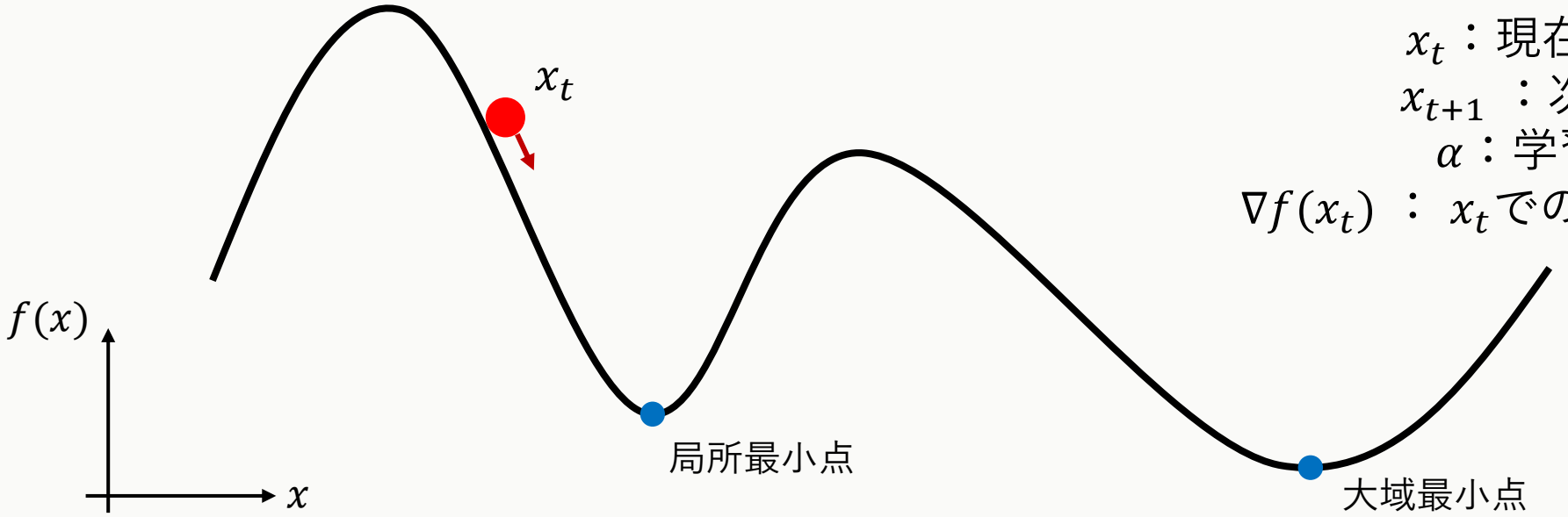
$$\hat{y} = W^{(4)}h^{(3)} + b^{(4)}$$
$$h^{(3)} = g(z^{(3)})$$
$$z^{(3)} = W^{(3)}h^{(2)} + b^{(3)}$$
$$h^{(2)} = g(z^{(2)})$$
$$z^{(2)} = W^{(2)}h^{(1)} + b^{(2)}$$
$$h^{(1)} = g(z^{(1)})$$
$$z^{(1)} = W^{(1)}x + b^{(1)}$$

深層学習における最適化のイメージ

- 関数の最小点(解)を探したい(関数を-1倍すれば最大点になる)
- ボールを転がすイメージ

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

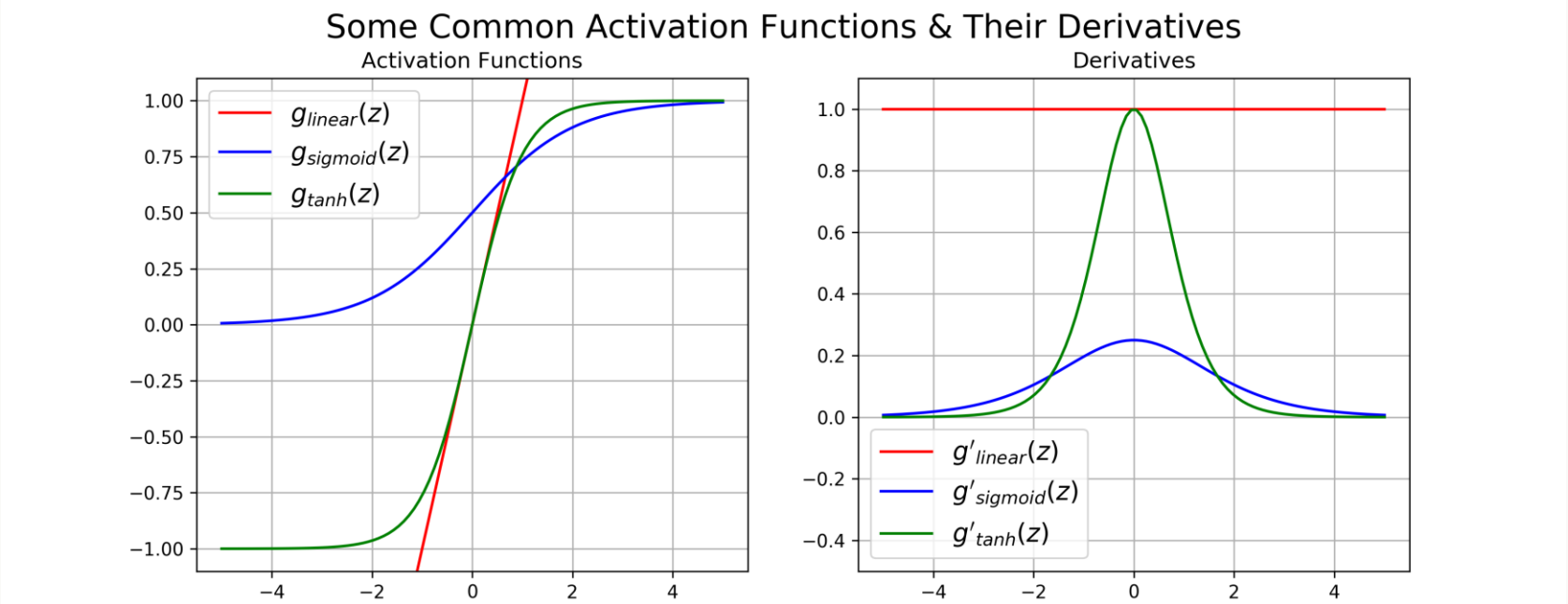
x_t：現在の点
x_{t+1}：次の点
α：学習率
$\nabla f(x_t)$：x_tでの関数fの勾配



深層学習の理解を深める

活性化関数の種々

- 活性化関数はNN中で微分される(=勾配が計算される)
  - 導関数の値域によって問題があるものも(勾配消失(Gradient vanishing)など)



正則化と正規化

- **正則化(Regularization)**：パラメータの偏りを減らし**過学習を防ぐ**
  - L1(Lasso), L2(Ridge)正則化など
- **正規化(Normalization)**：値のばらつきを減らし**学習を速める**
  - [0,1]への変換が一般的
  - 標準化(Standardization)：平均で引いて標準偏差で割る(中心に移動・拡大縮小)

参考： https://qiita.com/ryouka0122/items/a7fbad253680bb7f815e

データセットの使い方

- 基本的に**訓練用・テスト用の2つに分ける**
  - **テストで良い結果が出なければ意味がない**(汎化性能 ⇔ 過学習)
- データの分け方も様々
  - ホールドアウト(Hold-out)
  - ランダムサブサンプリング(Random subsampling)
  - 交差検証(Cross-validation)
  - 層化抽出(Stratified sampling)

参考文献

- “Solving the XOR problem using MLP.” Priyansh Kedia. https://medium.com/mlearning-ai/solving-the-xor-problem-using-mlp-83e35a22c96f, (最終参照日付 2022-4-27).
- “Derivation: Derivatives for Common Neural Network Activation Functions.” The Clever Machine. https://dustinstansbury.github.io/theclevermachine/derivation-common-neural-network-activation-functions, (最終参照日付 2022-4-27).