第20回医療健康データ科学Web セミナー（第3回機械学習Webセミナー）

深層学習の基礎

理化学研究所

hataya.tokyo

幡谷龍一郎

目次

目次

- **深層学習以前（ニューラルネットワークとは）**
- 深層学習
- 深層学習の実装

今年のノーベル物理学賞

The Nobel Prize in Physics 2024

John J. Hopfield

Geoffrey Hinton

*"for foundational discoveries and inventions that enable machine learning with **artificial neural networks**"*

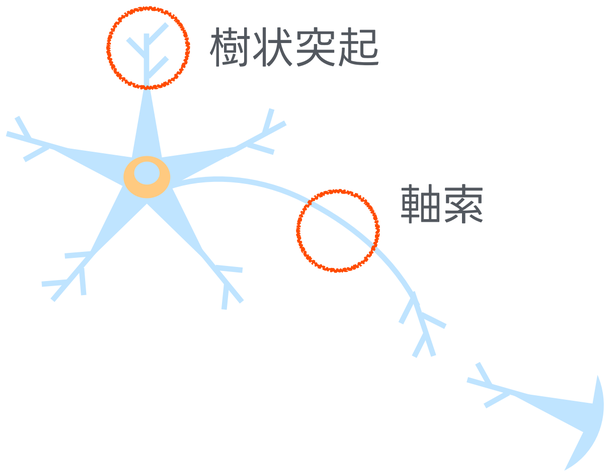
John Hopfield. Ill. Niklas Elmehed © Nobel Prize Outreach

Geoffrey Hinton. Ill. Niklas Elmehed © Nobel Prize Outreach

*ただし今回受賞対象となったタイプのニューラルネットは今回は取り扱わない

人工ニューロン

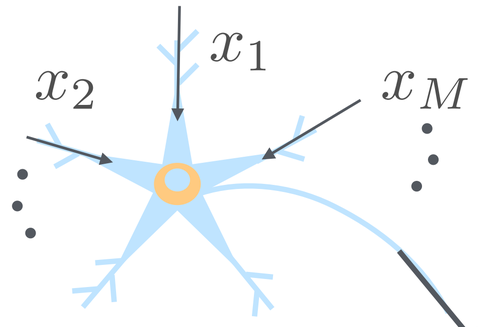
神経細胞



樹状突起への入力が閾値を超えた時だけ
軸索を通じて他の神経細胞に信号が伝達される
（全か無かの法則）

マッカロック・ピッツモデル [McCulloch&Pitts,43]

神経細胞の数理モデル（人工ニューロン）



$$\mathrm{H}\left(\sum_{i=1}^{M} w_i x_i + b\right)$$

w_i：重み　　b：バイアス

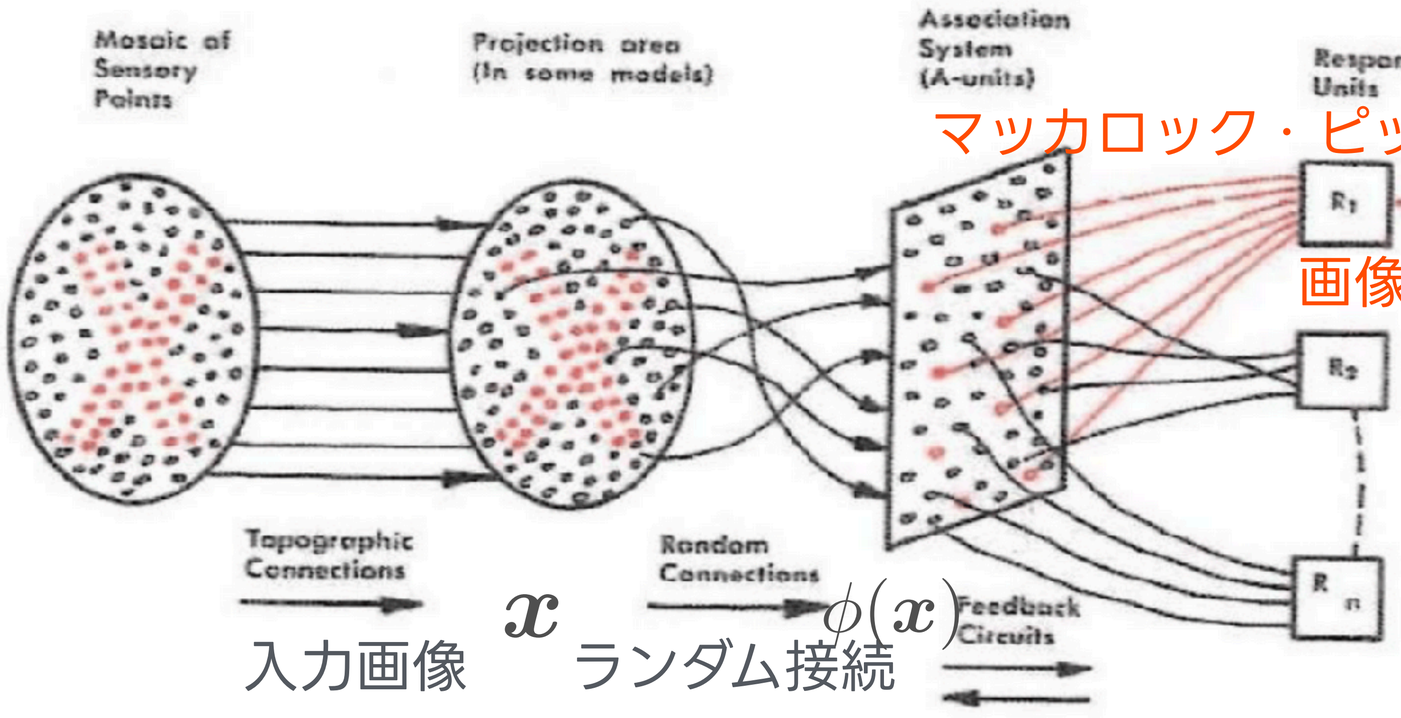
$$\mathrm{H}(x) = \begin{cases} 1 & (x \geq 0) \\ 0 & (x < 0) \end{cases}$$

$\sum_{i=1}^{M} w_i x_i$ が閾値 $-b$ を超えるとき出力は1

そうでなければ0

パーセプトロン [Rosenblatt,57]

マッカロック・ピッツモデルによってパターン認識を行う機械学習モデル



マッカロック・ピッツモデル

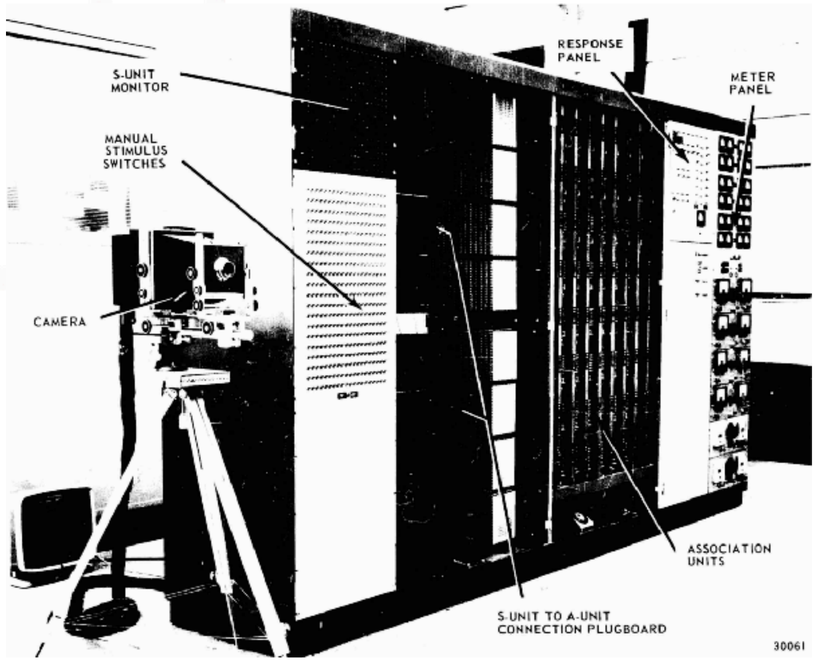
Output Imp. $\mathrm{H}(\boldsymbol{w}^\top \phi(\boldsymbol{x}) + b)$

画像はx?

$\boldsymbol{w}$ は誤答が少なくなるように更新

ハードウェアとして実装された

入力画像 $\boldsymbol{x}$ ランダム接続 $\phi(\boldsymbol{x})$



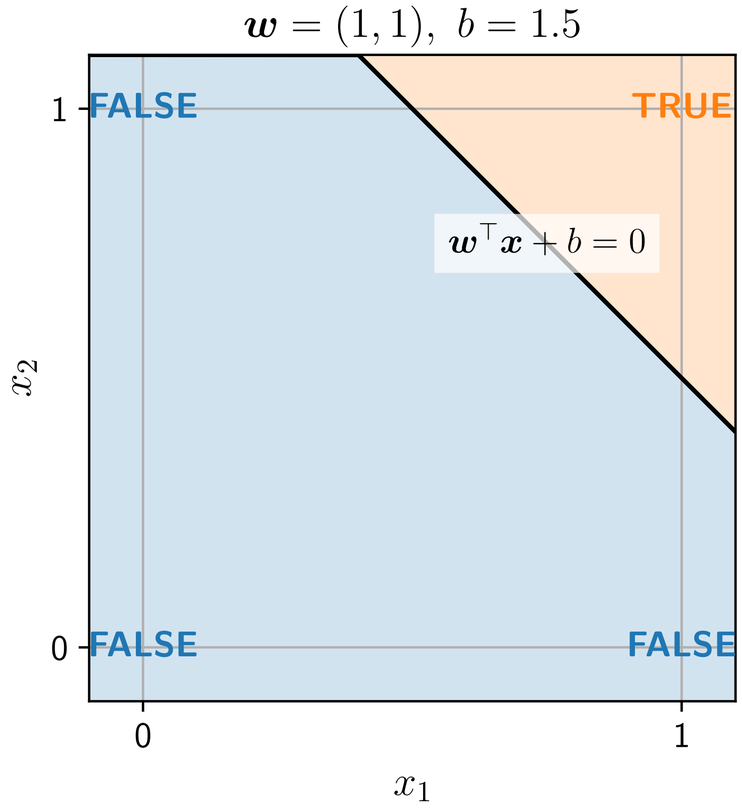
人工ニューロンの回路 → 人工ニューラルネットワーク

📼第一次ニューラルネットブームのきっかけ

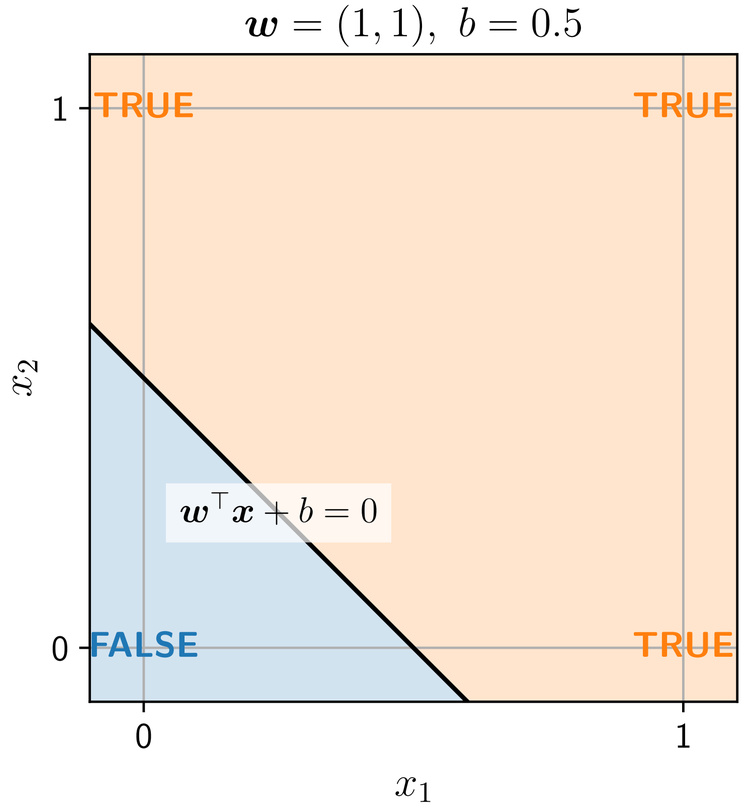
パーセプトロン [Rosenblatt,57]

マッカロック・ピッツモデル（パーセプトロン素子）は論理関数を表現できる

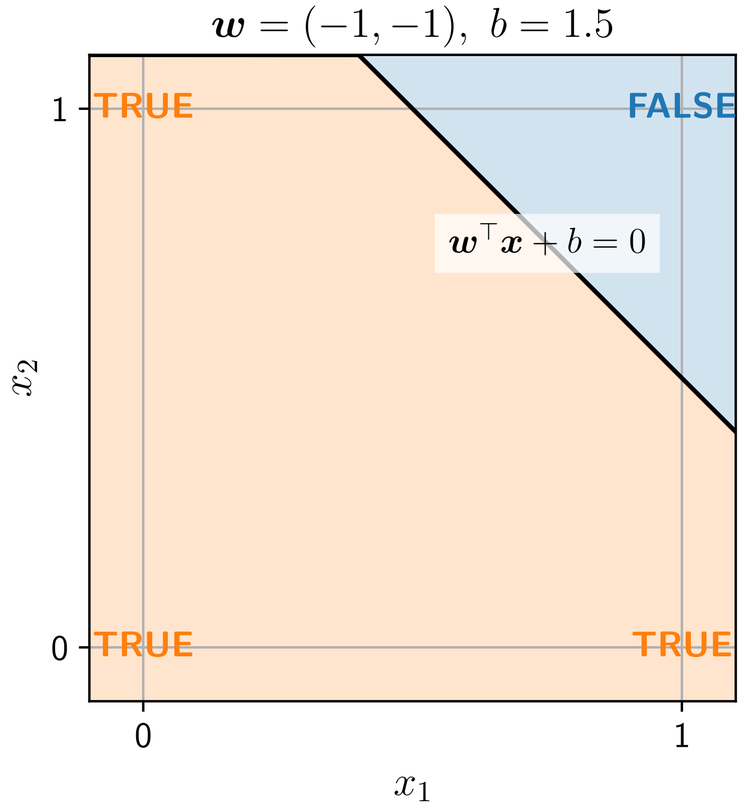
AND $x_1 \wedge x_2$



OR $x_1 \vee x_2$



NAND $\neg(x_1 \wedge x_2)$

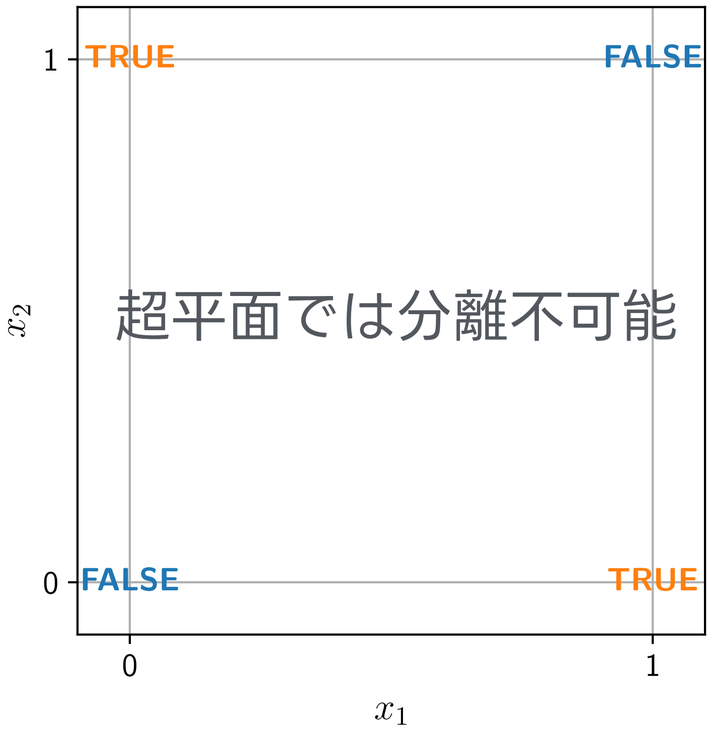


任意の論理関数はNANDの組合せで表現可能
→任意の論理関数は多層化したパーセプトロン素子により表現可能

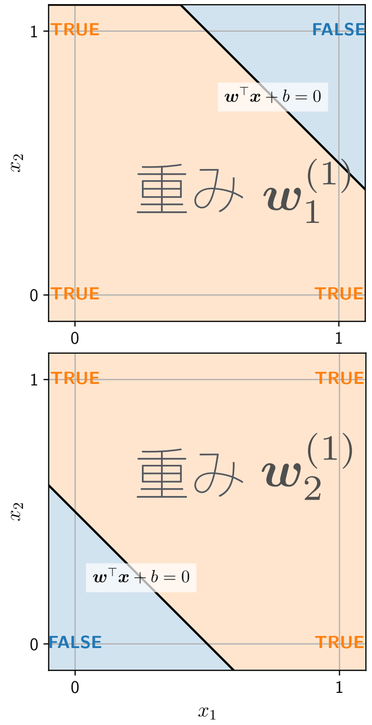
パーセプトロン [Rosenblatt,57]

マッカロック・ピッツモデル（パーセプトロン素子）は線形分離可能な問題しか扱えない

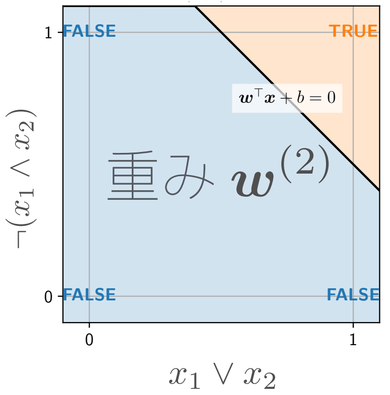
XOR $x_1 \oplus x_2$



多層にすれば可能 $x_1 \oplus x_2 = (x_1 \vee x_2) \wedge \neg(x_1 \wedge x_2)$

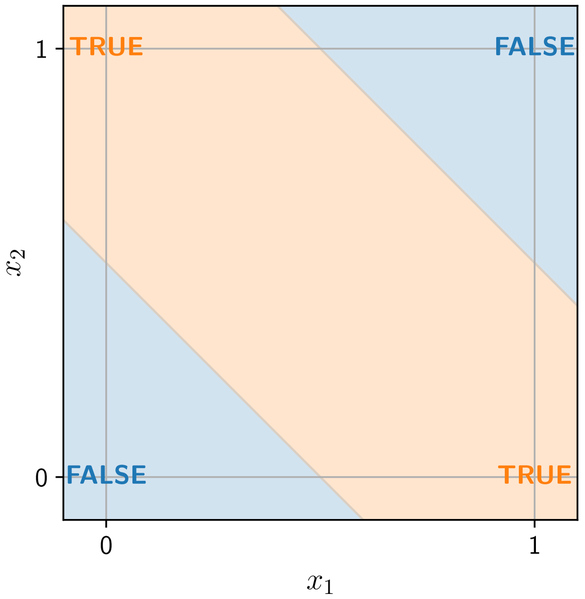


1層目



2層目

→



$$\mathrm{H}\left(\sum_{i=1}^{2} w_i^{(2)} \mathrm{H}(\boldsymbol{w}_i^{(1)} \boldsymbol{x} + b_i^{(1)}) + b^{(2)}\right)$$

- 1段だと線形分離不可能なことに注目が集まり第1次ニューラルネットブームは去った（らしい）
  - 多層の場合，最終層の重み（$\boldsymbol{w}^{(2)}, b^{(2)}$）しか更新できない問題も

誤差逆伝搬法（backpropagation）

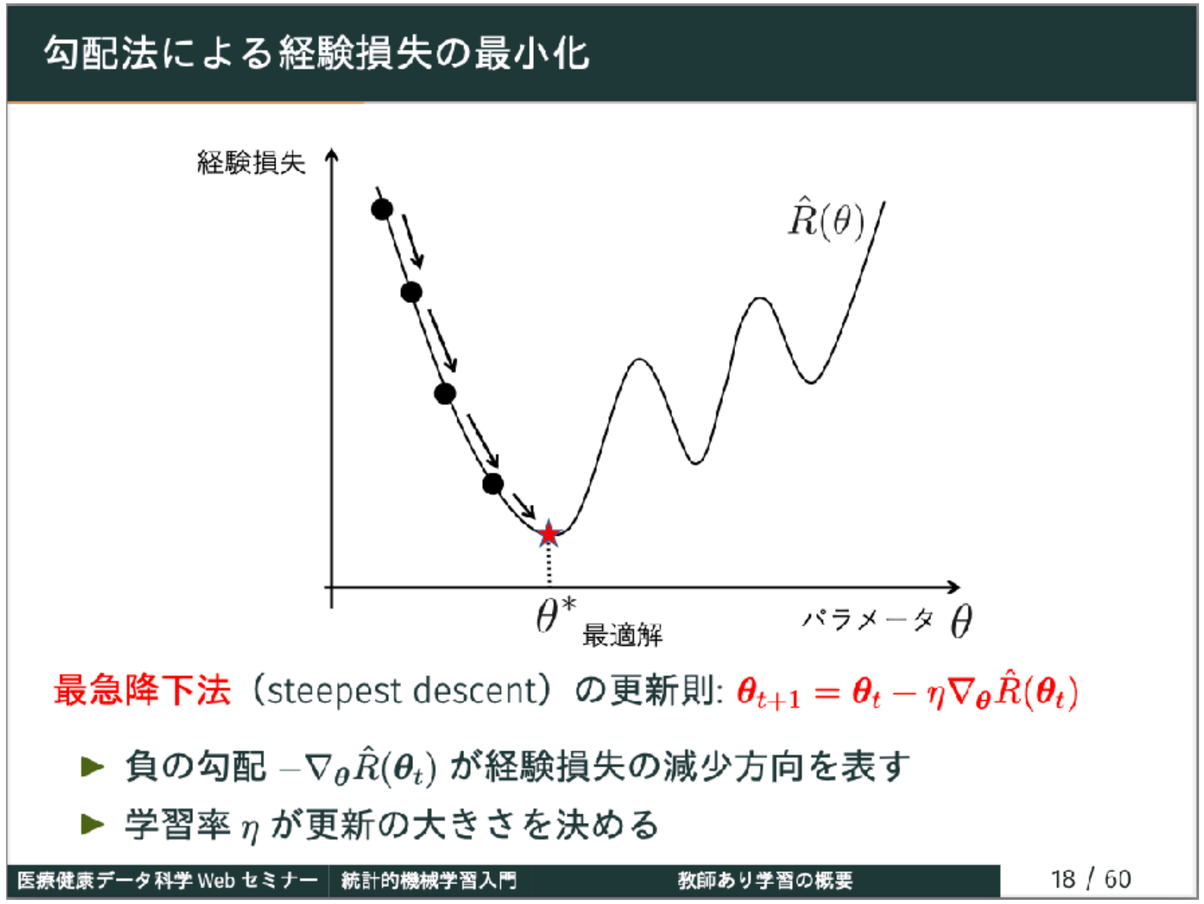
[Amari 67, Rumelhart+86]

Hintonも共著

2層のパーセプトロンでは2層目の重みしか更新できない

→ $\mathrm{H}(\sum_{i=1}^{2} w_i^{(2)} \mathrm{H}(\boldsymbol{w}_i^{(1)} \boldsymbol{x} + b_i^{(1)}) + b^{(2)})$ の H が微分可能であれば勾配法で更新可能

参考：



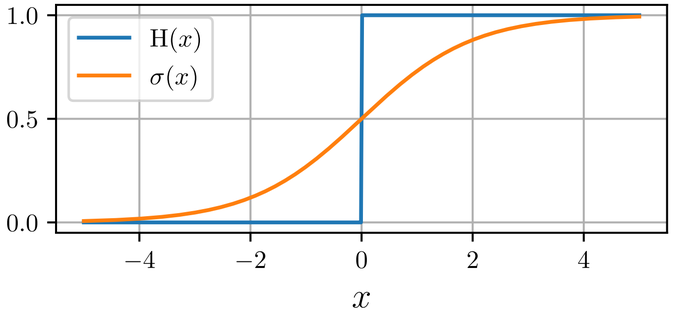
ここではθは$\boldsymbol{w}_1^{(1)}, \boldsymbol{w}_2^{(1)}, \boldsymbol{w}^{(2)}$に対応

松井先生の前回の講義

誤差逆伝搬法（backpropagation）

[Amari 67, Rumelhart+86]

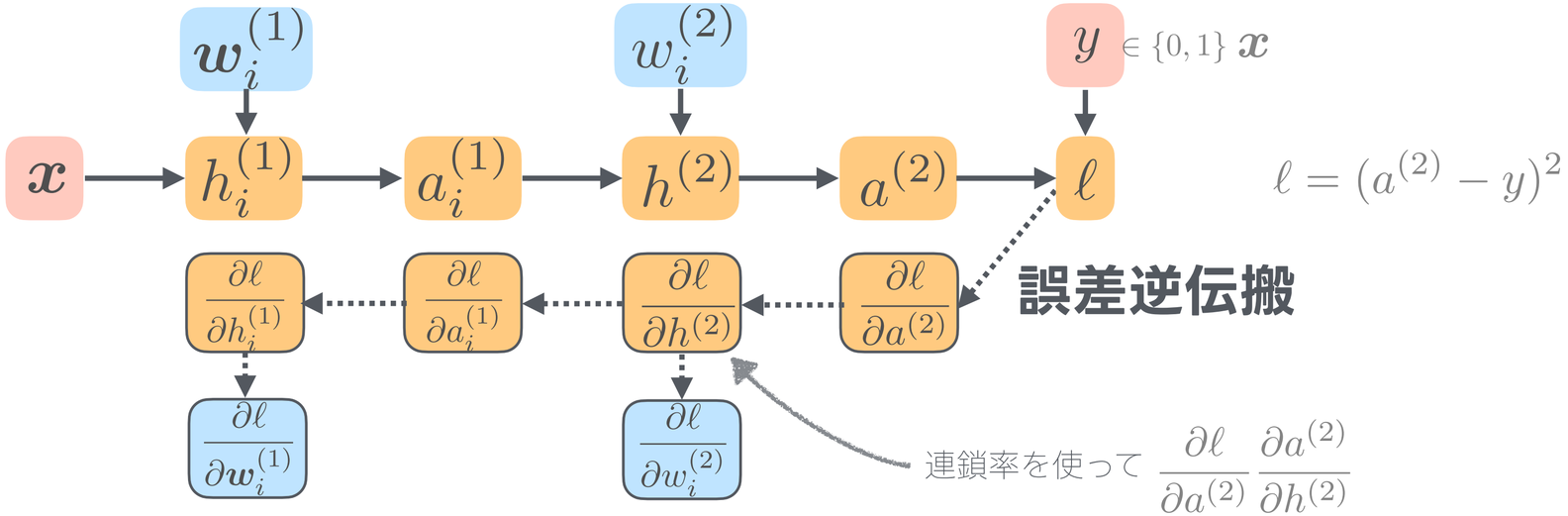
たとえば H をシグモイド関数 $\sigma(x) = \dfrac{1}{1+\exp(-x)}$



に置き換え $\quad a_i^{(1)} = \sigma(h_i^{(1)}), \quad h_i^{(1)} = \boldsymbol{w}_i^{(1)\top}\boldsymbol{x} + b_i^{(1)}$

$$a^{(2)} = \sigma(h^{(2)}), \quad h^{(2)} = \sum_{i=1}^{2} w_i^{(2)} a_i^{(1)} + b^{(2)}$$

$$\sigma\Big(\sum_{i=1}^{2} w_i^{(2)} \sigma(\boldsymbol{w}_i^{(1)}\boldsymbol{x} + b_i^{(1)}) + b^{(2)}\Big)$$



→ 多層のニューラルネットも学習可能に　📼第二次ニューラルネットブームのきっかけ

ここまでの一般化

1層目 $a_i^{(1)} = \sigma(h_i^{(1)}), \quad h_i^{(1)} = \boldsymbol{w}_i^{(1)\top}\boldsymbol{x} + b_i^{(1)}$

2層目 $a^{(2)} = \sigma(h^{(2)}), \quad h^{(2)} = \sum_{i=1}^{2} w_i^{(2)} a_i^{(1)} + b^{(2)}$

1層目 $\boldsymbol{a}^{(1)} = \eta(\boldsymbol{h}^{(1)}), \quad \boldsymbol{h}^{(1)} = \boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}$

$\boldsymbol{h}^{(1)} \in \mathbb{R}^{d_1}$　$\boldsymbol{x} \in \mathbb{R}^{d_0}$　$\boldsymbol{W}^{(1)} \in \mathbb{R}^{d_1 \times d_0}$, $\boldsymbol{b}^{(1)} \in \mathbb{R}^{d_1}$

重み行列　**バイアス**

まとめて**パラメータ**

ベクトルの要素ごとに施す

非線形関数（例：シグモイド関数）

活性化関数と呼ばれる

2層目 $\boldsymbol{a}^{(2)} = \eta(\boldsymbol{h}^{(2)}), \quad \boldsymbol{h}^{(2)} = \boldsymbol{W}^{(2)}\boldsymbol{a}^{(1)} + \boldsymbol{b}^{(2)}$

$\boldsymbol{W}^{(2)} \in \mathbb{R}^{d_2 \times d_1}$, $\boldsymbol{b}^{(2)} \in \mathbb{R}^{d_2}$

$\vdots$

l層目 $\boldsymbol{a}^{(l)} = \eta(\boldsymbol{h}^{(l)}), \quad \boldsymbol{h}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$

$\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$, $\boldsymbol{b}^{(l)} \in \mathbb{R}^{d_l}$

$\vdots$

L層目 $\boldsymbol{h}^{(L)} = \boldsymbol{W}^{(L)}\boldsymbol{a}^{(L-1)} + \boldsymbol{b}^{(L)}$

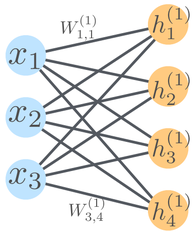
$\boldsymbol{W}^{(L)} \in \mathbb{R}^{d_L \times d_{L-1}}$, $\boldsymbol{b}^{(L)} \in \mathbb{R}^{d_L}$

重み行列が

「普通」の行列のとき，

線形層（アフィン層）

全結合層と呼ばれる



線形層+活性化関数を重ねたもの

多層パーセプトロン（MLP）

フィードフォワードネットワーク

と呼ばれる

ここまでの一般化

1層目 $\boldsymbol{a}^{(1)} = \eta(\boldsymbol{h}^{(1)}),$ $\boldsymbol{h}^{(1)} = \boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}$

$\boldsymbol{h}^{(1)} \in \mathbb{R}^{d_1}$ $\boldsymbol{x} \in \mathbb{R}^{d_0}$ $\boldsymbol{W}^{(1)} \in \mathbb{R}^{d_1 \times d_0},\ \boldsymbol{b}^{(1)} \in \mathbb{R}^{d_1}$

ベクトルの要素ごとに施す
非線形関数（例：シグモイド関数）
活性化関数と呼ばれる

重み行列 **バイアス**
まとめて**パラメータ**

2層目 $\boldsymbol{a}^{(2)} = \eta(\boldsymbol{h}^{(2)}),$ $\boldsymbol{h}^{(2)} = \boldsymbol{W}^{(2)}\boldsymbol{a}^{(1)} + \boldsymbol{b}^{(2)}$

$\boldsymbol{W}^{(2)} \in \mathbb{R}^{d_2 \times d_1},\ \boldsymbol{b}^{(2)} \in \mathbb{R}^{d_2}$

$\vdots$

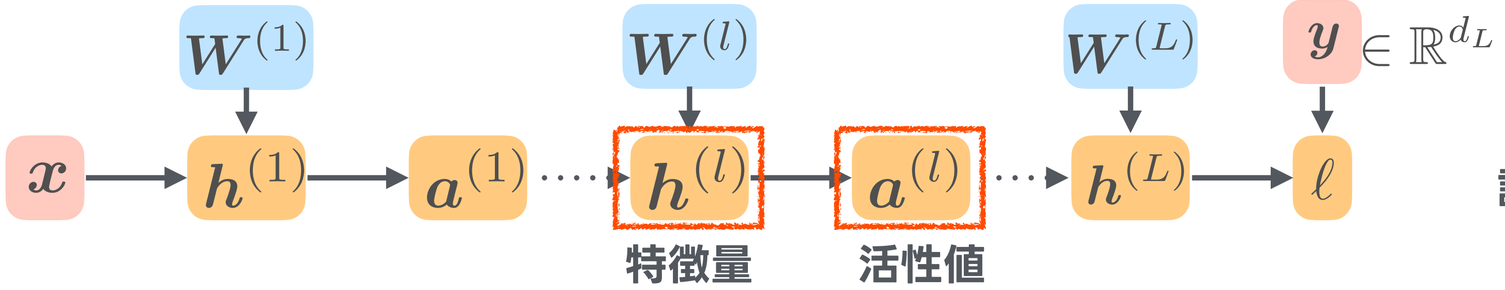
l層目 $\boldsymbol{a}^{(l)} = \eta(\boldsymbol{h}^{(l)}),$ $\boldsymbol{h}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$

$\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}},\ \boldsymbol{b}^{(l)} \in \mathbb{R}^{d_l}$

$\vdots$

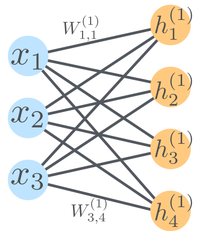
L層目 $\boldsymbol{h}^{(L)} = \boldsymbol{W}^{(L)}\boldsymbol{a}^{(L-1)} + \boldsymbol{b}^{(L)}$

$\boldsymbol{W}^{(L)} \in \mathbb{R}^{d_L \times d_{L-1}},\ \boldsymbol{b}^{(L)} \in \mathbb{R}^{d_L}$



誤差逆伝播が可能

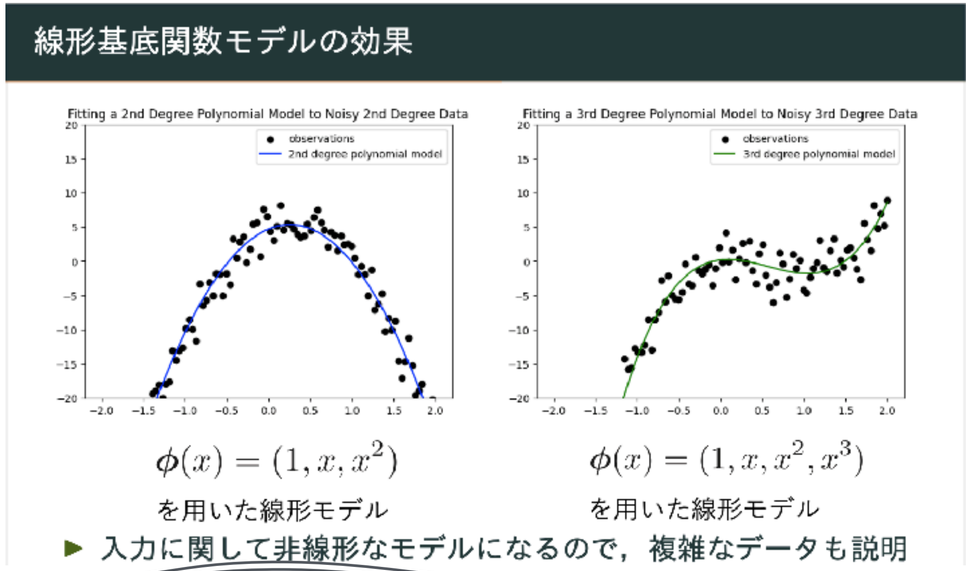
重み行列・バイアスが
普通の行列のとき，
線形層（アフィン層）
全結合層と呼ばれる



線形層+活性化関数を重ねたもの
多層パーセプトロン（MLP）
フィードフォワードネットワーク
と呼ばれる

基底関数モデルとの関係

参考：



松井先生の前回の講義

基底関数モデル：

$$f(\boldsymbol{x}) = \sum_{i=1}^{M} w_i \phi_i(\boldsymbol{x})$$

基底関数は**固定**

2層ニューラルネット：

$$f(\boldsymbol{x}) = \sum_{i=1}^{M} w_i^{(2)} \eta(\boldsymbol{w}_i^{(1)\top} \boldsymbol{x} + b_i^{(1)})$$

基底関数がデータに**適応可能**

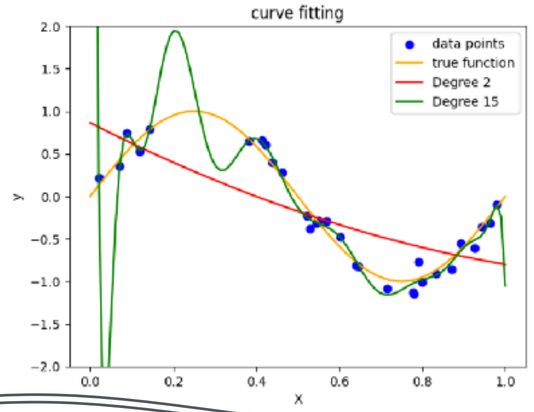
重み共有によるパラメータ削減

パラメータが多いと過剰適合が生じる可能性がある

参考：

正則化つき線形回帰 i

Recall 単純な誤差最小化は過剰適合を招く（→ モデル選択）



松井先生の前回の講義

$$f(\boldsymbol{x}) = \sum_{i=1}^{M} w_i \phi_i(\boldsymbol{x})$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{M} w_i^{(2)} \eta(\boldsymbol{w}_i^{(1)\top} \boldsymbol{x} + b_i^{(1)})$$

ニューラルネットはパラメータが多いため過剰適合しやすい

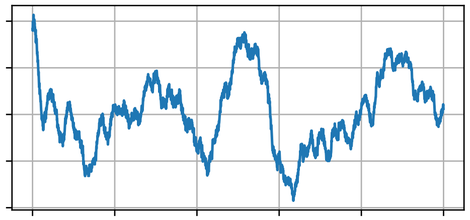
容易にデータセットを「丸暗記」してしまう

→ **データの構造**を利用し重みを共有することでパラメータ数を削減

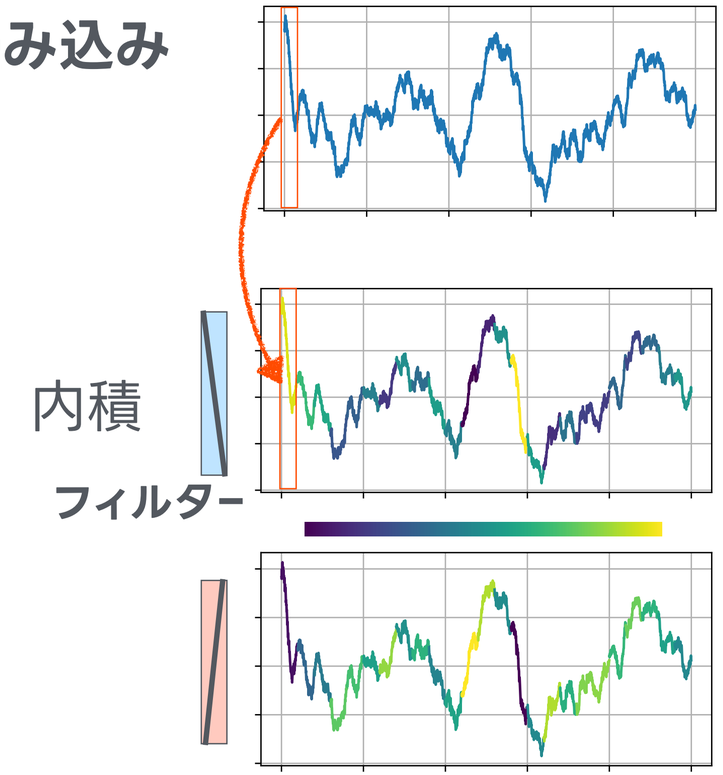
畳み込みニューラルネット [LeCun+89]

画像：

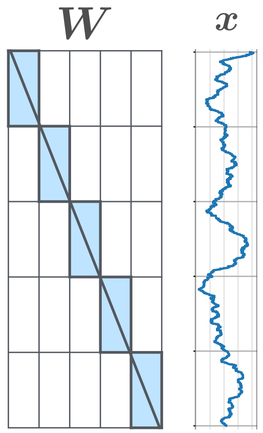
音声：



畳み込み



行列積で表現可能



共有された  のみ学習

（ほかはゼロ）

→ パラメータ削減

（データの構造を捉えているので
学習もしやすい）

（詳細は次回以降）

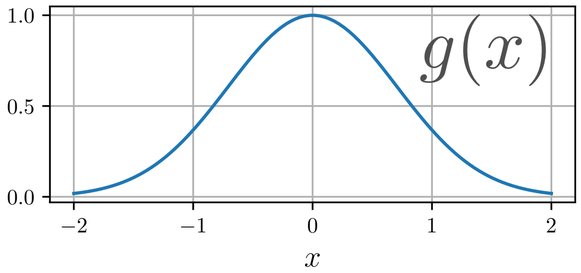
📼画像認識・音声認識に利用されてきた

ニューラルネットの万能近似性 [Cybenko89] etc.,

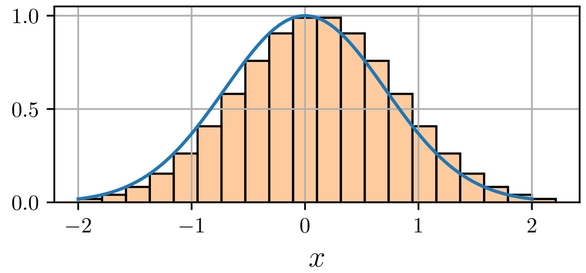
1変数入力1変数出力の2層ニューラルネット $f(x) = \sum_{i=1}^{M} w_i^{(2)} \sigma(w_i^{(1)} x + b_i^{(1)})$ は

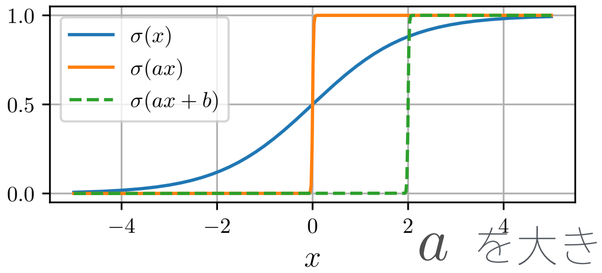
任意の1変数関数 $g(x)$ を近似可能

証明の方針



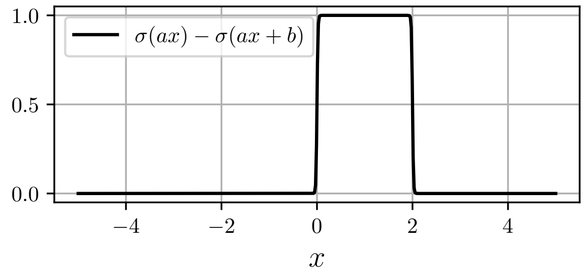
$g(x)$ は区分定数関数によって近似可能





区分定数関数はシグモイド関数2つで近似可能

a を大きく取る



区分定数関数の高さを決める

$$f(x) = \sum_{i=1}^{M} w_i^{(2)} \sigma(w_i^{(1)} x + b_i^{(1)})$$

区分定数関数を作る

任意の1変数関数 $g(x)$ を近似可能

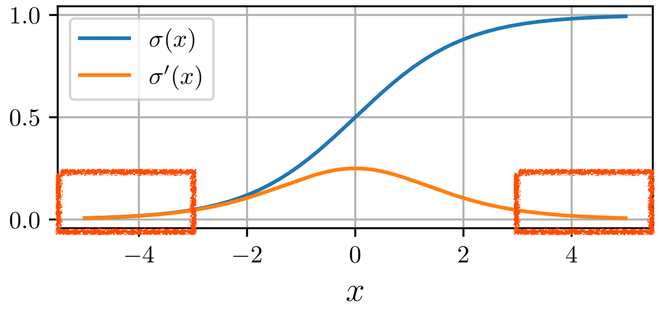
ニューラルネットの層を増やす

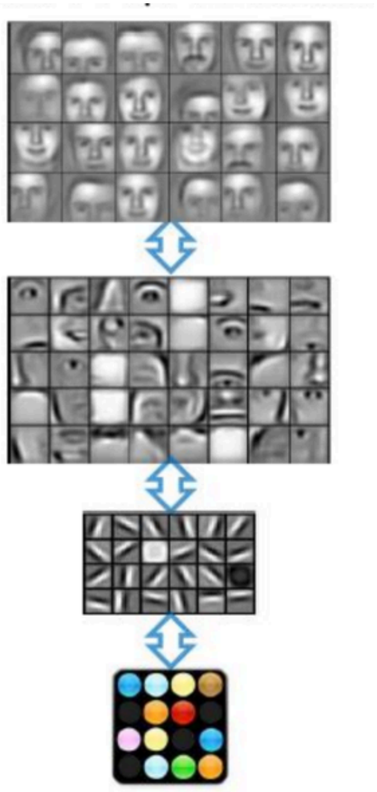
ニューラルネットの層を増やす（深くする）と
データを階層的に処理できるようになる

より複雑なタスクを行うためには深くする必要

深くすると誤差逆伝搬した勾配が消失・発散して
学習が失敗してしまう

一因：





カテゴリ　高レベル

パーツ

エッジ

ピクセル　低レベル

📼第二次ニューラルネットブームの終焉

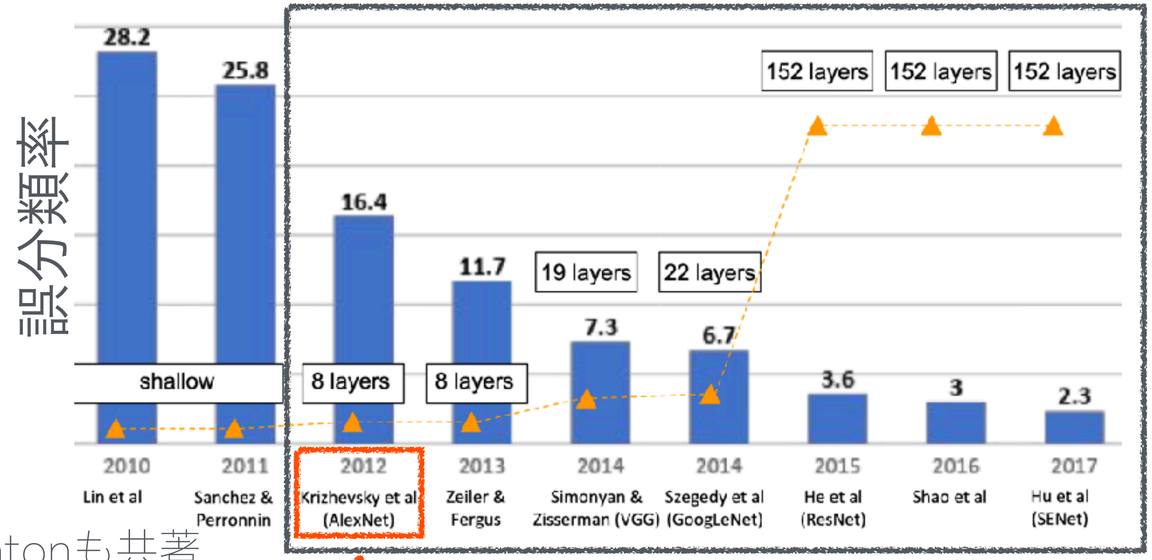
Adapted from [Lee,10]

目次

- 深層学習以前
- **深層学習（深層学習を可能にする技術）**
- 深層学習の実装

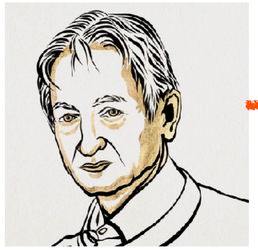
深層学習の勃興

ImageNetデータセットの
1000カテゴリ 120万枚の画像を分類するコンペ



ImageNet中の画像

Hintonも共著



深層畳み込みニューラルネットワーク

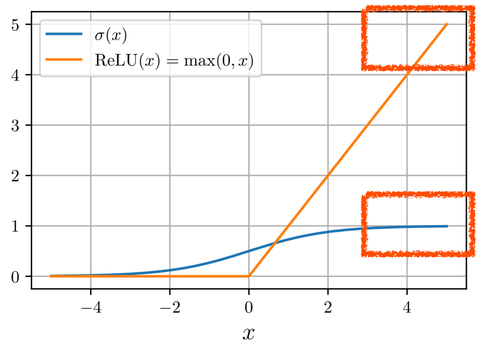
AlexNetの成功理由

- 活性化関数をReLUへの変更
- 過剰適合を抑制する「ドロップアウト」の導入
- データ量の増加+計算機の進化（GPU）

Image: adapted from the course material of Stanford CS231n

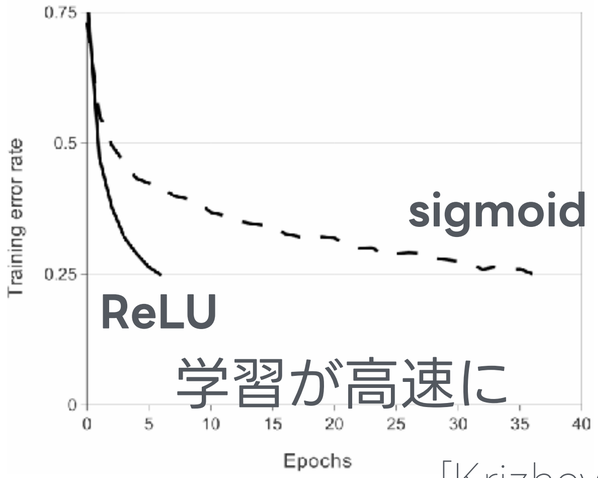
AlexNetの成功理由

ReLU活性化関数　$\mathrm{ReLU}(x) = \max(0, x)$



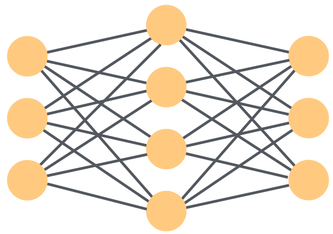
勾配が消失しない

勾配が消失



[Krizhevsky+12]

ドロップアウト



学習時にランダムに各層の出力ベクトルの要素を0に

→ 些細な特徴を学習しにくくなる（過剰適合の抑制）

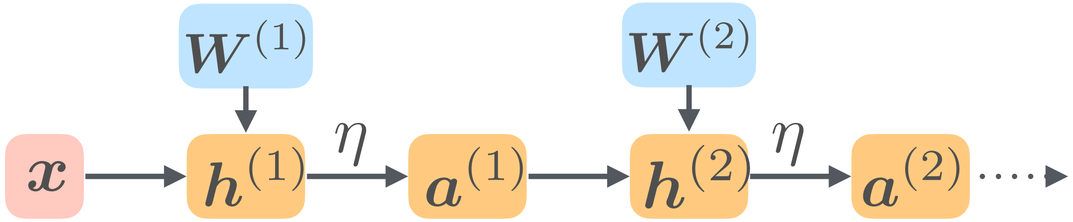
データ量の増加+計算機の進化（GPU）

インターネットの普及によりデータの収集が容易に

深層学習に現れる行列積の並列計算がGPUにより高速に

パラメータの初期化

パラメータはランダムに初期化



$\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \ldots,$ の分散が大きく変化しないように初期化したい

Kaiming法（He法とも）　活性化関数がReLUのとき用いる

$$W_{ij}^{(l)} \sim \mathcal{N}(0, \frac{2}{d_{l-1}}) \text{ あるいは } \mathcal{U}(-\sqrt{\frac{6}{d_{l-1}}}, \sqrt{\frac{6}{d_{l-1}}})$$

d_{l-1} ：層への入力ベクトルの次元

実際　$h_j^{(l)} = \sum_{i=1}^{d_{l-1}} W_{ji} a_i^{(l-1)}$

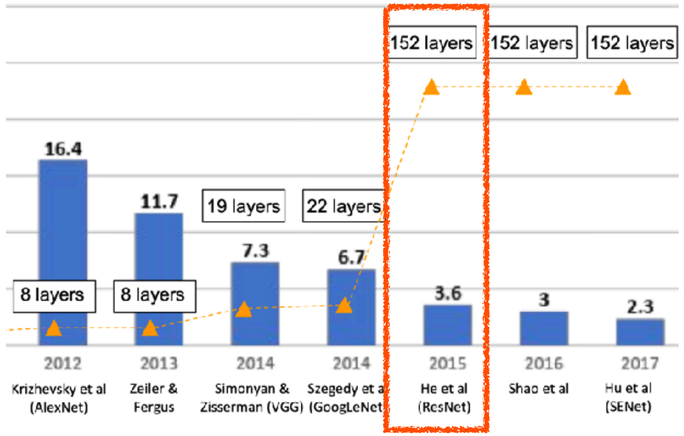
$$\begin{aligned}
\mathbb{V}(h_j^{(l)}) &= d_{l-1}\mathbb{V}(W_{ji}^{(l)} a_i^{(l-1)}) \\
&= d_{l-1}\mathbb{E}(W_{ji}^{(l)2})\mathbb{E}(a_i^{(l-1)2}) \\
&= d_{l-1}\mathbb{V}(W_{ji}^{(l)})\frac{\mathbb{V}(h_i^{(l-1)})}{2}
\end{aligned}$$

$W_{ji}^{(l)}, a_i^{(l)}$ は独立，$\mathbb{V}(x) = \mathbb{E}(x^2) - \mathbb{E}(x)^2$

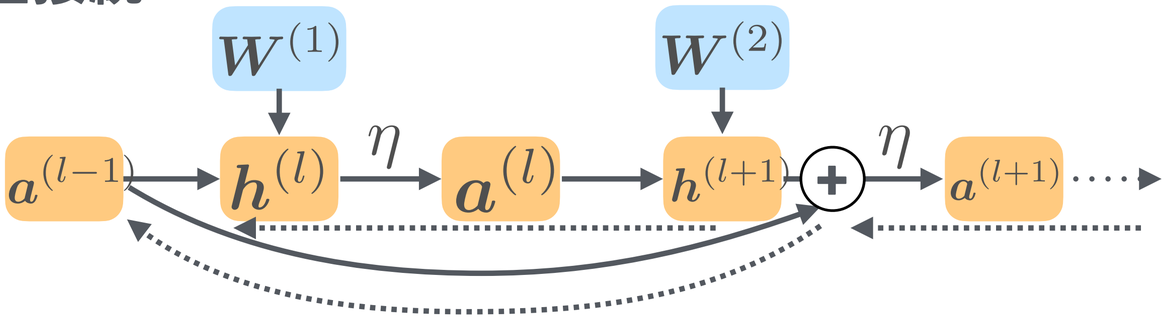
ReLUなので右側だけ

ResNet

ResNetは100層以上の深さを実現

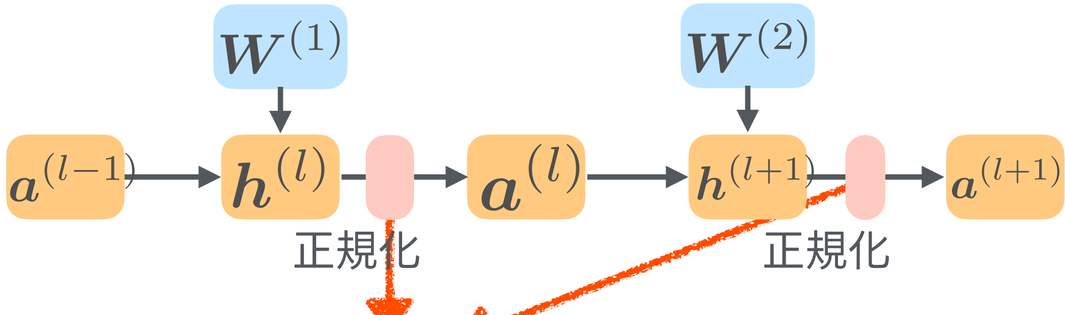


残差接続



勾配の消失を防ぐ
→ 100～1000層も可能に

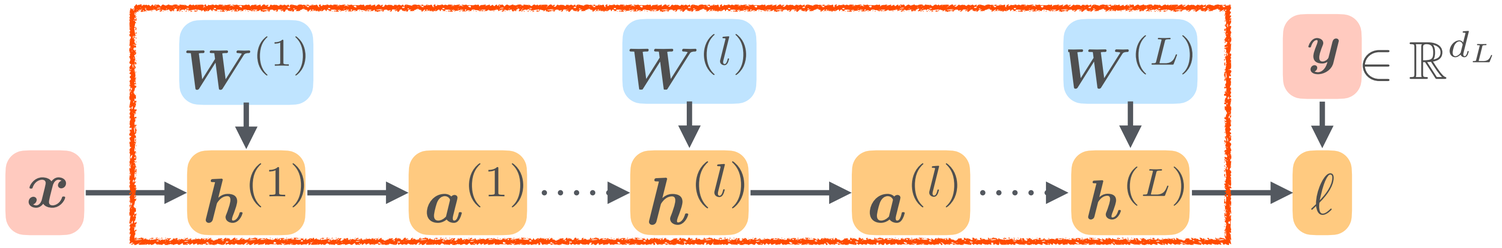
特徴量の正規化



同じ平均・分散
→ 学習の安定

→残差接続・特徴量の正規化は最新のニューラルネットでも利用されている技術

確率的勾配降下法（SGD）



$\boldsymbol{\theta} = (\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \ldots \boldsymbol{W}^{(L)}, \boldsymbol{b}^{(L)})$ として $f_{\boldsymbol{\theta}} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_L}$ と表記

深層学習では一般的

バッチ学習

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \frac{1}{N} \sum_{\substack{(\boldsymbol{x}_n, \boldsymbol{y}_n) \\ \in \mathcal{D}}} L(f_{\boldsymbol{\theta}}(\boldsymbol{x}_n), \boldsymbol{y}_n)$$

（ミニバッチを用いた）
確率的勾配降下法

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \frac{1}{M} \sum_{\substack{(\boldsymbol{x}_n, \boldsymbol{y}_n) \\ \in \mathcal{B}}} L(f_{\boldsymbol{\theta}}(\boldsymbol{x}_n), \boldsymbol{y}_n)$$

$\mathcal{B} \subset \mathcal{D}, |\mathcal{B}| = M$ は大きさMのミニバッチでランダムに選択

全データにアクセスせずに済む
並列化しやすい
ランダム性が正則化に

確率的勾配降下法

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L(f_{\boldsymbol{\theta}}(\boldsymbol{x}_n), \boldsymbol{y}_n)$$

n は毎回ランダムに選択

α ：学習率

$\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$：データセット

確率的勾配降下法（SGD）

通常SGDは以下とともに用いる　　$\boldsymbol{g}_t = \nabla_{\boldsymbol{\theta}} \frac{1}{M} \sum_{\substack{(\boldsymbol{x}_n, \boldsymbol{y}_n) \\ \in \mathcal{B}}} L(f_{\boldsymbol{\theta}_t}(\boldsymbol{x}_n), \boldsymbol{y}_n)$　とする

モメンタム

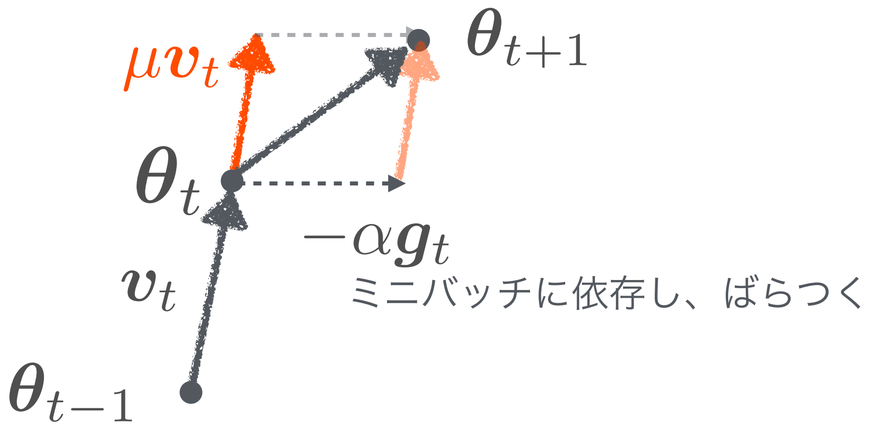
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \boldsymbol{g}_t + \mu \boldsymbol{v}_t$$

$$\boldsymbol{v}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}$$

SGDを安定化させ収束性を改善する補正

実質的な学習率を大きくする効果も

$\mu = 0.9$ 程度に設定される



重み減衰

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha(\boldsymbol{g}_t + \lambda \boldsymbol{\theta}_t)$$

以下の正則化付きの損失を用いていることと等価

$$L'(f_{\boldsymbol{\theta}}(\boldsymbol{x}_n), \boldsymbol{y}_n) = L(f_{\boldsymbol{\theta}}(\boldsymbol{x}_n), \boldsymbol{y}_n) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2$$

より発展的な確率的勾配降下法

Adam [Kingma&Ba 14]

$$\theta_{t+1,i} = \theta_{t,i} - \alpha \frac{\hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}} + \varepsilon}$$

勾配の1次モーメント（モメンタム）

$$\hat{m}_{t,i} = \frac{m_{t,i}}{1-\beta_1^t} \qquad m_{t,i} = \beta_1 m_{t-1,i} + (1-\beta_1) g_{t,i}$$

モーメント推定値の偏差を補正

勾配の2乗モーメント

$$\hat{v}_{t,i} = \frac{v_{t,i}}{1-\beta_2^t} \qquad v_{t,i} = \beta_2 v_{t-1,i} + (1-\beta_2) g_{t,i}^2$$

モーメント推定値の偏差を補正

これまで勾配が大きかった成分iは控えめに更新
小さかった成分iは積極的に更新

→ 学習率をパラメータごとに適応的に調整している

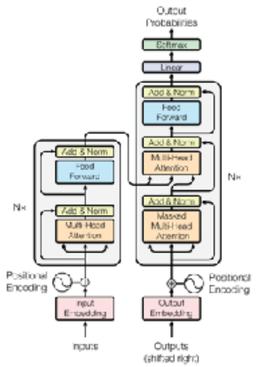
AdamはSGDよりも安定した学習が可能で，広く用いられている

現代の深層学習はAdamに過剰適合しているとも…

なぜ深層学習を使うのか

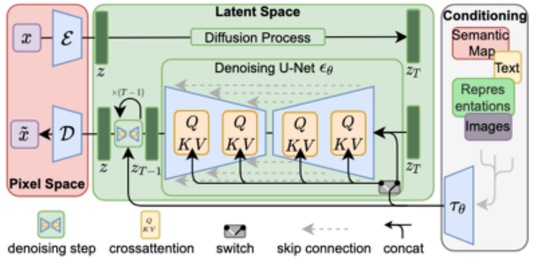
（データが十分にあれば）様々な関数を表現できる

言語モデル



これまでの単語列を入力，
次の単語を出力とする関数

画像生成モデル



画像のノイズを少し除去する関数
（何度も繰り返してノイズから画像をつくる）

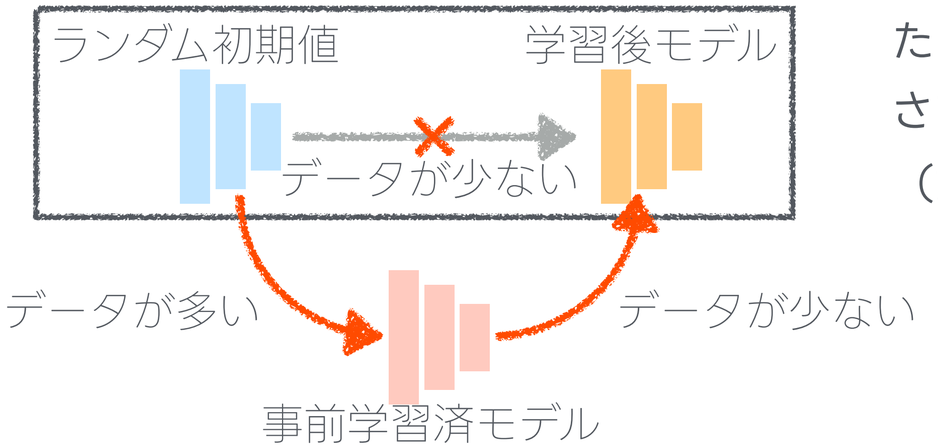
ニューラルネットは各層は簡単な計算なので大規模化しやすい
層をブロックのように組み合わせることで様々な関数をモデル化できる

Images from [Vaswani+ 17, Rombach+ 21]

なぜ深層学習を使うのか

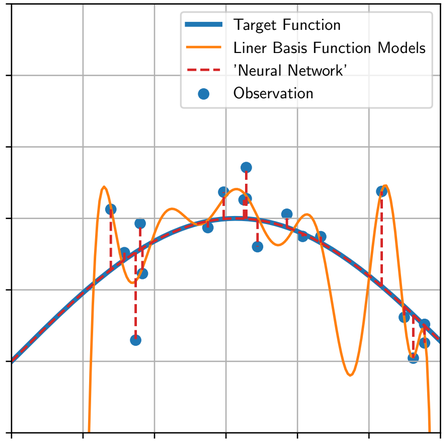
転移学習

事前学習済みのパラメータを初期値とすることで，少ないデータでも学習可能



たとえばImageNetで事前学習した画像認識モデルは
さまざまな画像認識タスクに転移されている
（ただし転移学習がネガティブに働く場合もあるので注意）

良性過剰適合



ニューラルネットは学習データを「丸暗記」してもよく汎化する

→ 十分にパラメータがあるので，目標の関数をよく近似しつつ
観測ノイズはスパイクで表現している

図はイメージです

目次

実装

- ウェブ上で実行できるGoogle colabで共有しました
- https://bit.ly/3DhALFH

文献紹介

- 岡谷貴之「深層学習」講談社，2022.
  - 深層学習を概観する和書

- 斎藤康毅「ゼロから作るDeep Learning」2016.
  - Pythonを使って文字通りゼロから深層学習の理解を目指すシリーズ

- Zhang, Lipton, Li, and Smola, “Dive into Deep Learning,” 2023.
  - オンライン版にはPyTorchなどのコードも付属　https://d2l.ai

- PyTorch tutorial https://pytorch.org/tutorials/index.html
  - 日本語翻訳（更新は数年前に止まっているが基本的には問題ないはず）　https://yutaroogawa.github.io/pytorch_tutorials_jp/