

A I 解析入門

渡辺英治（TSBセンター・AI解析室）

2025年2月5日（水）

講師自己紹介

渡辺英治（わたなべえいじ）

専門：神経科学

テーマ：脳の構成論



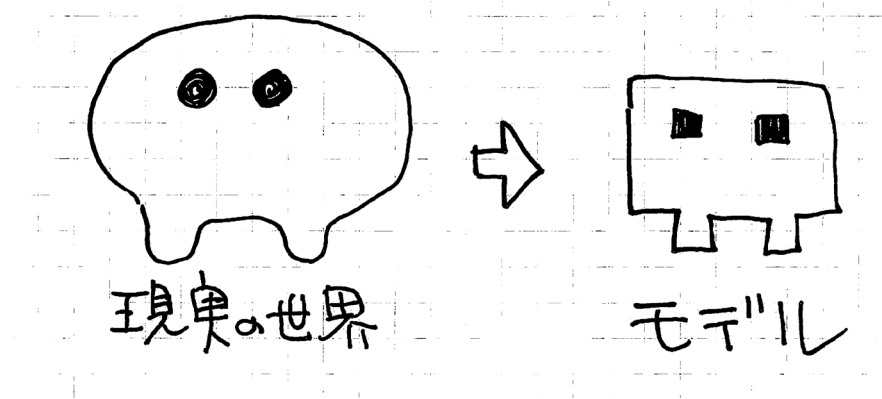
“Neuronist”

講義の概要

1. 人工ニューロンを通して深層学習の本質を理解する
2. 現在のA Iブームの起爆剤となりノーベル物理学賞の対象にもなったAlexNetを作って動かして体感する

モデル

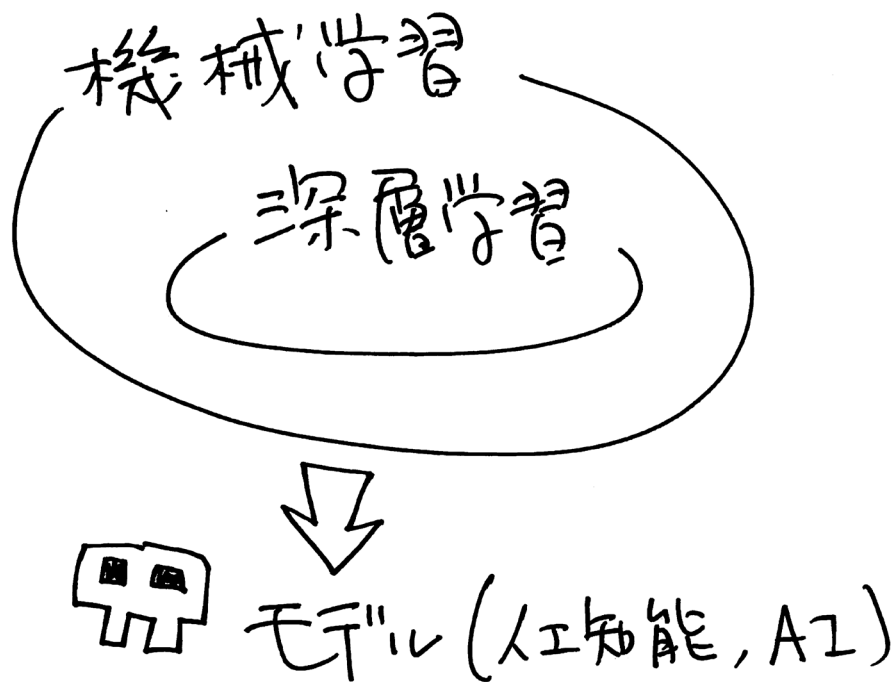
1. 現実世界の複雑な事象や系を単純化あるいは抽象化して表現したものがモデル



2. 機械学習、深層学習はモデルを作る作業
3. 関数は数式やコードで明確に定義された入出力関係を表現する。モデルは関数を拡張した概念。ニューラルネットワークは巨大な関数。関数であるが故にコンピュータで動かすことができる

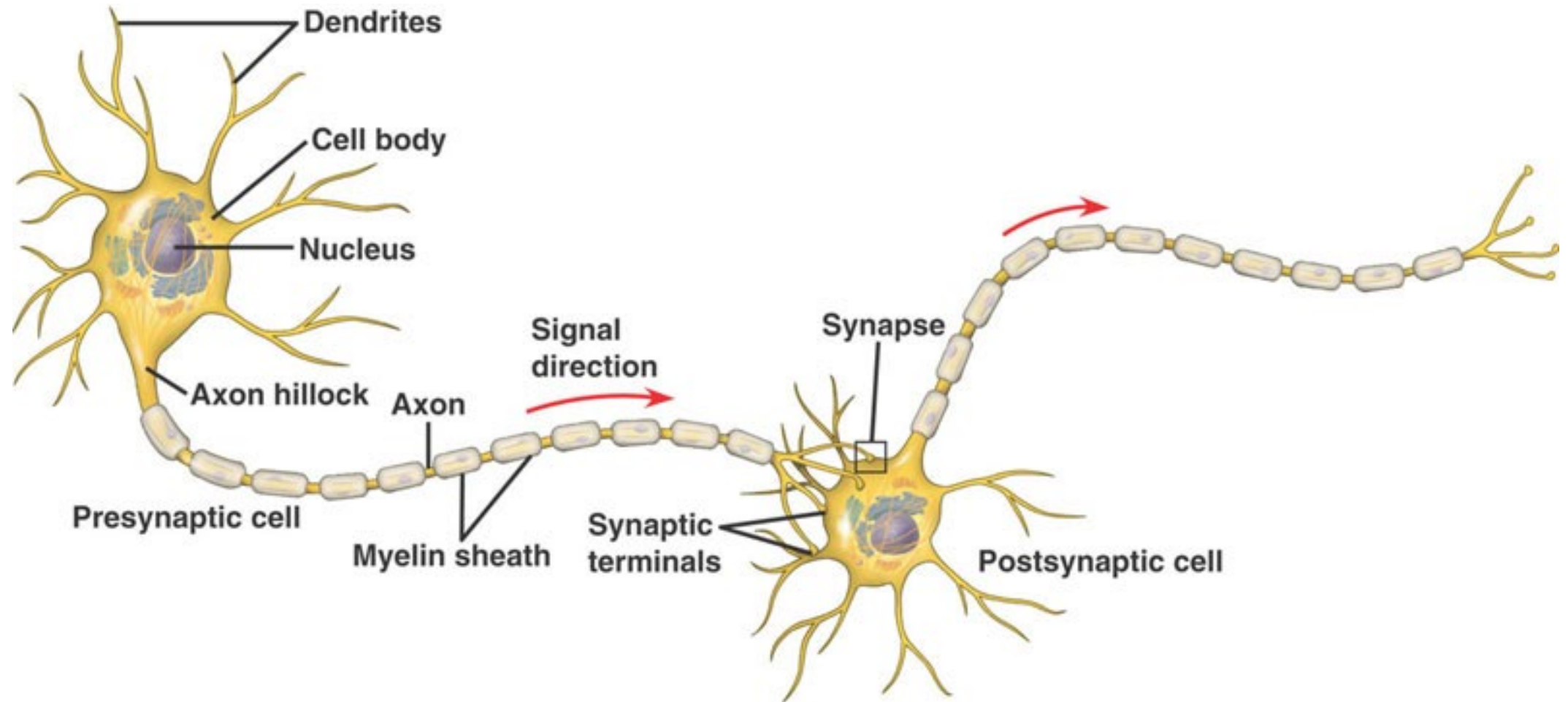
機械学習と深層学習

1. モデル獲得の過程をコンピュータで自動化したものが機械学習
2. 人工ニューロンを使った機械学習が深層学習



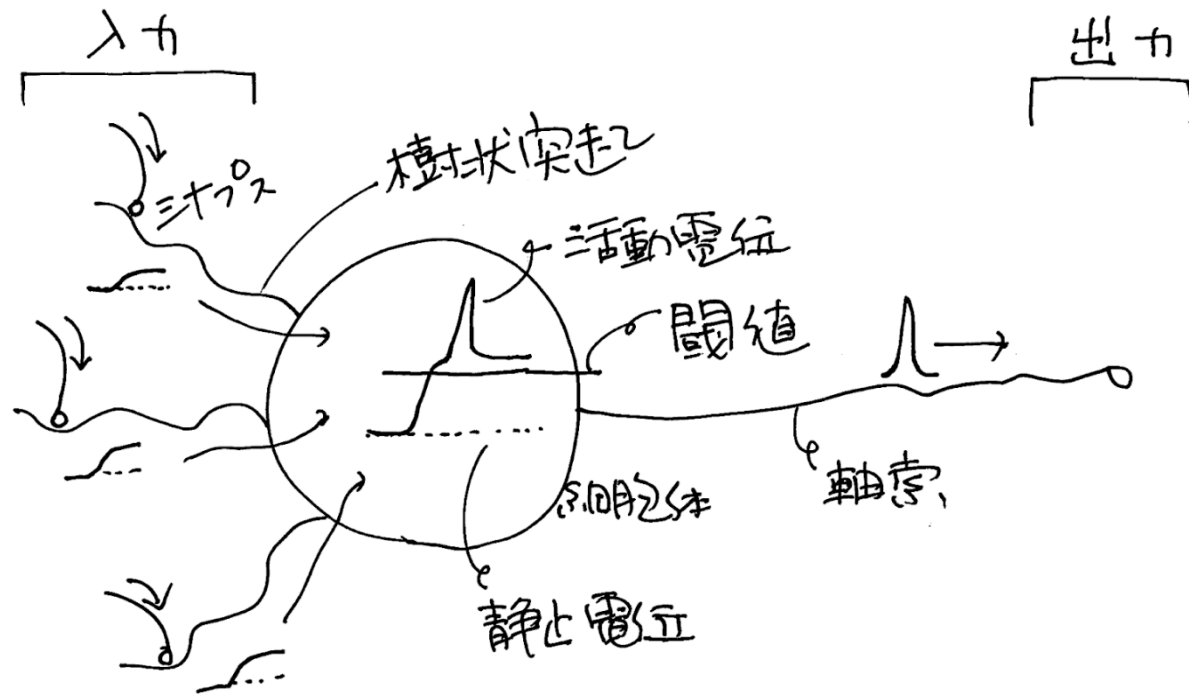
人工ニューロン

神經細胞 (Neuron)



神経細胞 (Neuron)

- 1) 細胞として独立した単位である
- 2) 他の神経細胞から情報を受ける樹状突起と他の神経細胞へ情報を送り出す軸索を細胞体から伸ばしている
- 3) 細胞間（シナプス）では化学的神経伝達で情報が伝わる
- 4) 細胞膜内外には電位差（膜電位）。膜電位は細胞が静止状態のときは細胞外が正、細胞内が負である
- 5) 化学的神経伝達では相手の膜電位を上昇させるか下降させる
- 6) 膜電位の正負が反転し、電位差が拡大して閾値を超えると活動電位が発生する
- 7) 活動電位は軸索を伝わりシナプスに到達すると化学的神経伝達を引き起こす



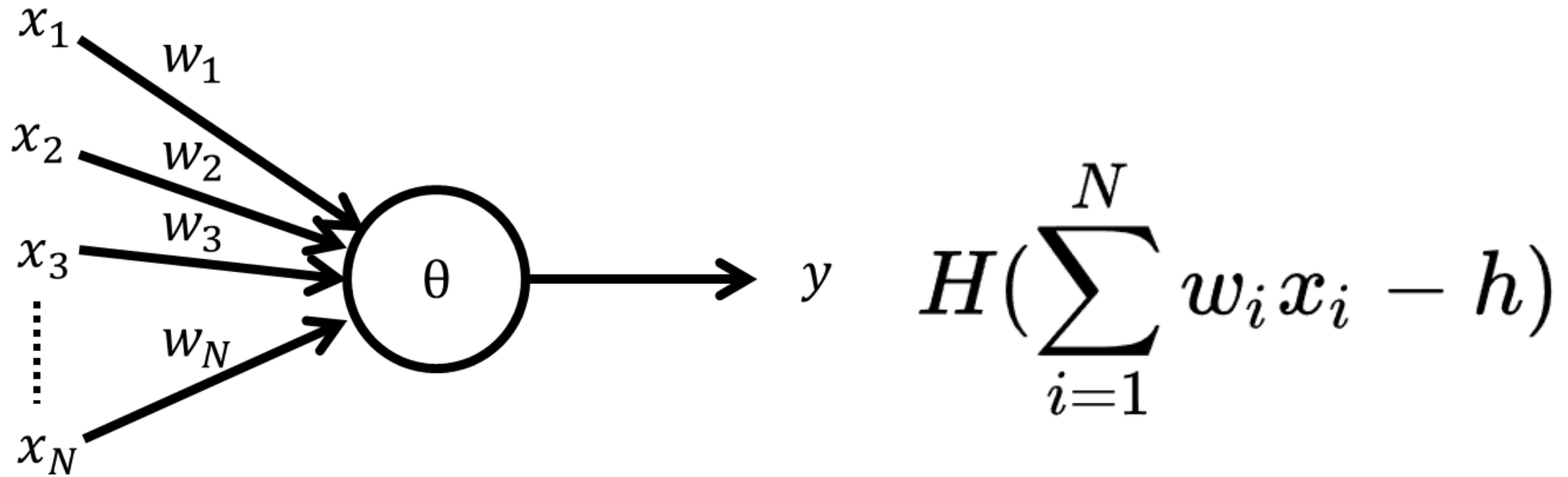
人工ニューロンの発明



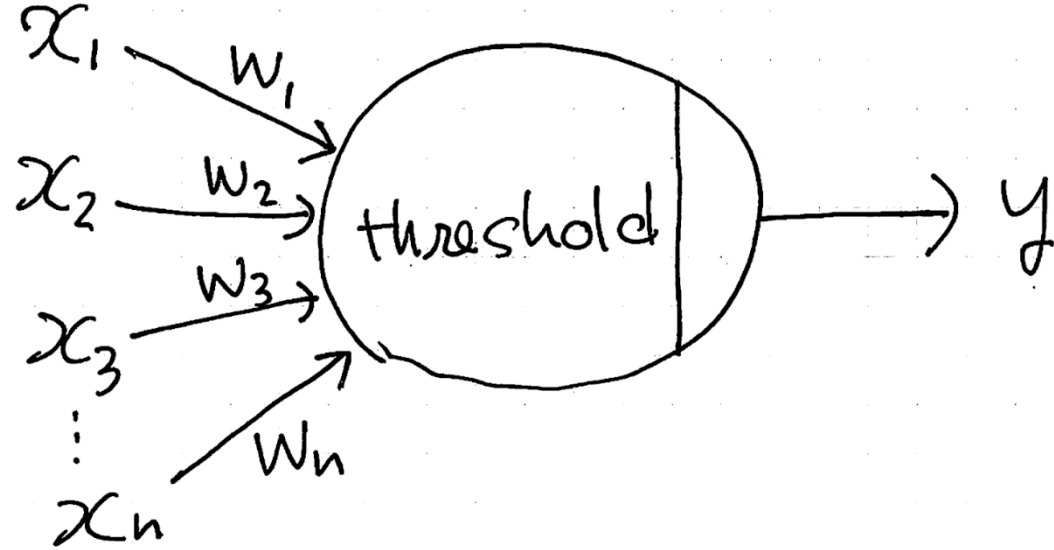
Warren S. McCulloch
神経生理学者



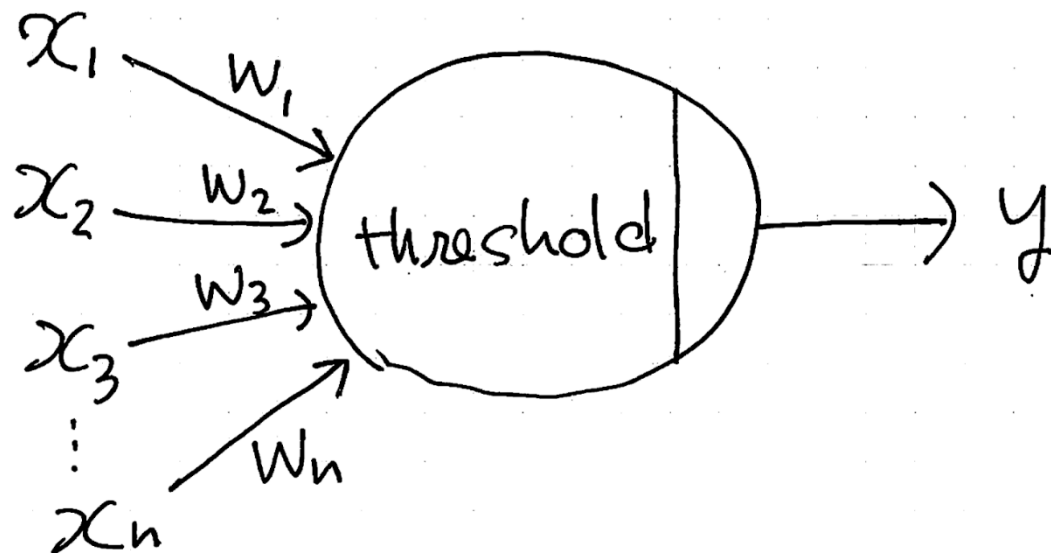
Walter Pitts
論理科学者／数学者



McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 7:115 - 133.



- 1) 荷重 $\mathbf{w} = (w_1, w_2, \dots, w_n)$
- 2) 入力 $\mathbf{x} = (x_1, x_2, \dots, x_n)$
- 3) 荷重と入力の内積 $\mathbf{w} \cdot \mathbf{x} = w_1 * x_1 + \dots w_n * x_n$
- 4) 内積と閾値との比較を行う $\mathbf{w} \cdot \mathbf{x} - \text{threshold}$
- 5) 出力を計算する $y = H(\mathbf{w} \cdot \mathbf{x} - \text{threshold})$
- 6) H は階段方程式で、 $()$ 内が正のときに1, 負のときに0を出力する



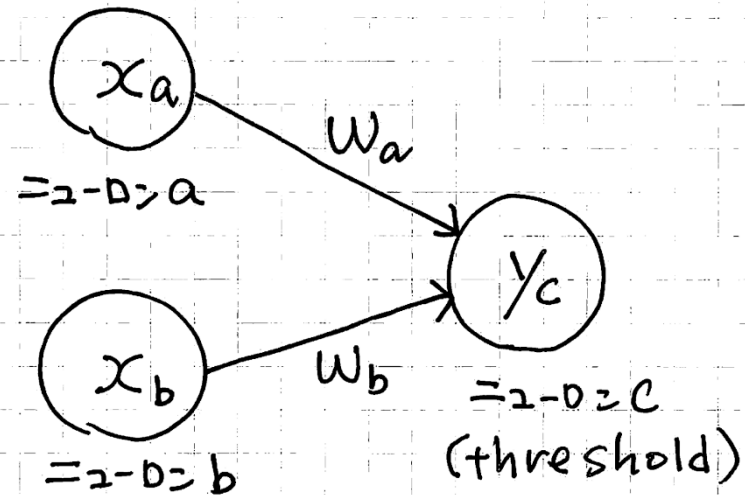
- 1) 荷重 $\mathbf{w} = (w_1, w_2, \dots, w_n)$
- 2) 入力 $\mathbf{x} = (x_1, x_2, \dots, x_n)$
- 3) 荷重と入力の内積 $\mathbf{w} \cdot \mathbf{x} = w_1 * x_1 + \dots + w_n * x_n$
- 4) 内積と閾値との比較を行う $\mathbf{w} \cdot \mathbf{x} - \text{threshold}$
- 5) 出力を計算する $y = H(\mathbf{w} \cdot \mathbf{x} - \text{threshold})$
- 6) H は階段方程式で、 $()$ 内が正のときに1, 負のときに0を出力する
- 7) 以上が「形式ニューロン」である。ときにはパーセプトロンと呼ばれる
- 8) パーセプトロンはRosenblattによる提案されたニューラルネットワークである
- 9) \mathbf{w} とthresholdを人工ニューロンのパラメータと呼ぶ (学習対象)

パーセプトロンの能力

基本的な論理演算は4つある。AND演算、OR演算、NOT演算、XOR演算である。パーセプトロンはこの4つの論理演算を表現できる。

a	b	c
0	0	0
1	0	0
0	1	0
1	1	1

AND



$$y_c = H(x_a * w_a + x_b * w_b - \text{threshold})$$

例えば、 w_a 、 w_b 、 threshold の値を、それぞれ5、5、9
としてみる。

$$y_c = H(5x_a + 5x_b - 9)$$

パーセプトロンの能力

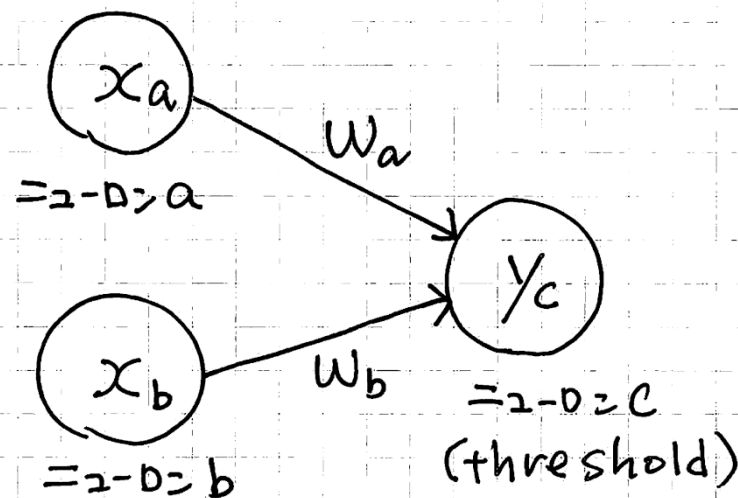
AND演算のモデルが出来た！！

w_a 、 w_b 、thresholdの組み合わせは無数にある

モデルに対するパラメータは無数にあることを理解しておきたい

a	b	c
0	0	0
1	0	0
0	1	0
1	1	1

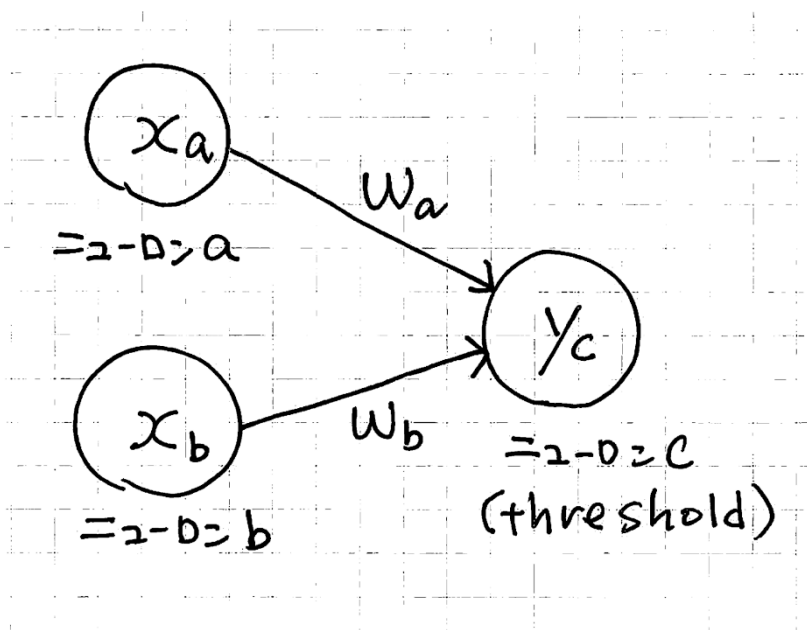
AND



パーセプトロンの能力

a	b	c
0	0	0
1	0	1
0	1	1
1	1	1

OR



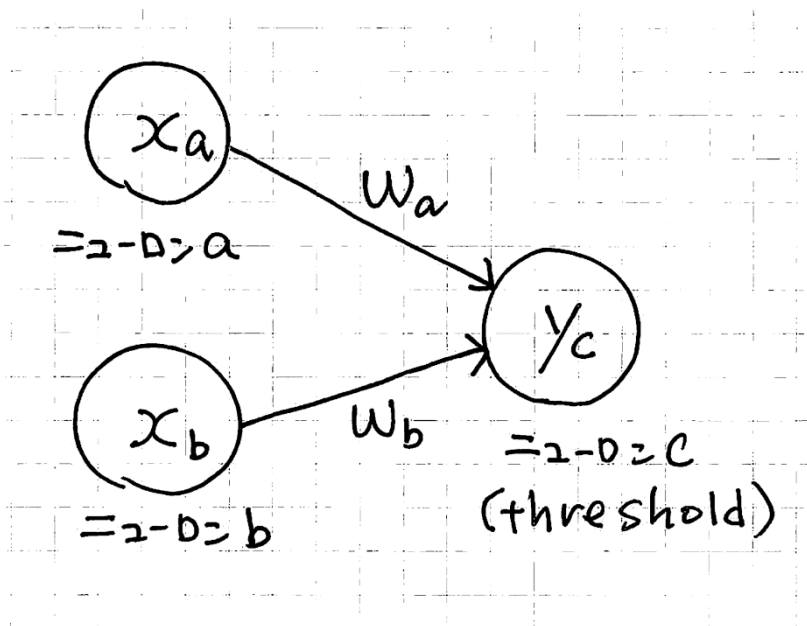
$$y_c = H(x_a * w_a + x_b * w_b - \text{threshold})$$

例えば、 w_a 、 w_b 、thresholdの値を、 ???

パーセプトロンの能力

a	b	c
0	0	1
1	0	1
0	1	1
1	1	0

NAND



$$y_c = H(x_a * w_a + x_b * w_b - \text{threshold})$$

例えば、 w_a 、 w_b 、 threshold の値を、それぞれ -5 、 -5 、 -9 としてみる（ANDの逆）。

$$y_c = H(-5x_a - 5x_b + 9)$$

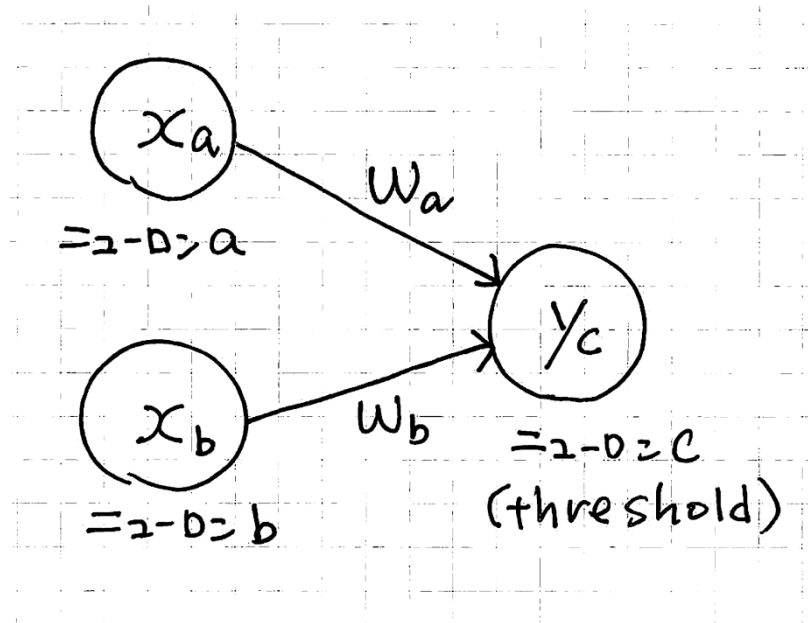
パーセプトロンの能力

NAND演算のモデルが出来たがthresholdが負になった！！

thresholdの値が負になり活動電位を発するような生物学的な神経細胞はこの世には存在しない。ここが数式として表現された架空のニューロンの面白いところである。細胞は数式になった瞬間、現実の物理世界の制約から解放されて自由になるのである。生物学的な神経細胞のモデルとしては失格であるが、機能的な自由度は格段に上がることになる。この自由度もニューラルネットワークの大きな特質のひとつである。

a	b	c
0	0	1
1	0	1
0	1	1
1	1	0

NAND



パーセプトロンの能力

a	b	c
0	0	0
1	0	1
0	1	1
1	1	0

XOR

a	b	h ₁	h ₂	c
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

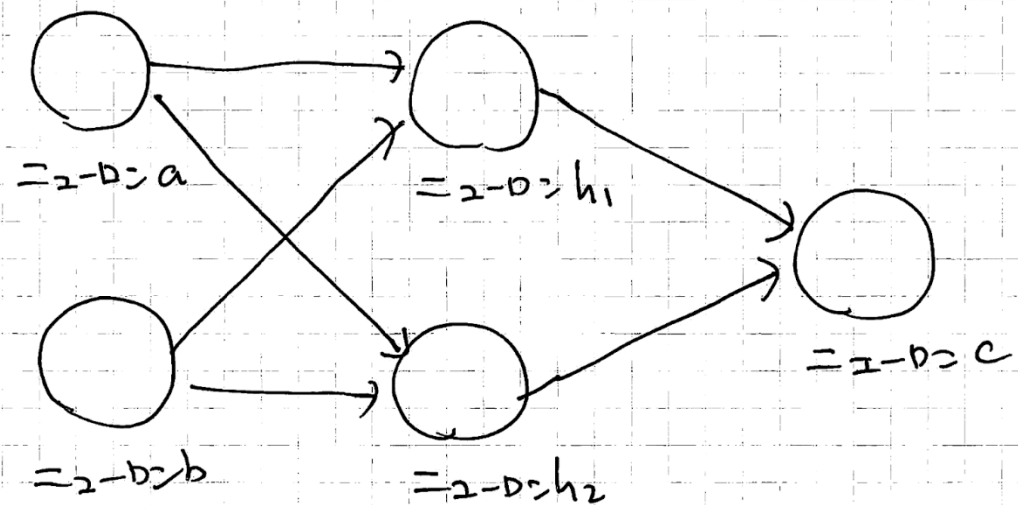
XOR

ニューロンa、bとニューロンcの間に、もう2つ隠れニューロンとしてh₁、h₂を追加すれば実現できる。ニューロンh₁、h₂ともにニューロンa、bの両方からの入力を受け、h₁がNAND演算として働き、h₂がOR演算として働き、そしてニューロンh₁、h₂から入力を受けるニューロンcがAND演算として働けばいい。

パーセプトロンの能力

a	b	h ₁	h ₂	c
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

XOR



ニューロンa、bとニューロンcの間に、もう2つ隠れニューロンとしてh1、h2を追加すれば実現できる。ニューロンh1、h2ともにニューロンa、bの両方からの入力を受け、h1がNAND演算として働き、h2がOR演算として働き、そしてニューロンh1、h2から入力を受けるニューロンcがAND演算として働けばいい。

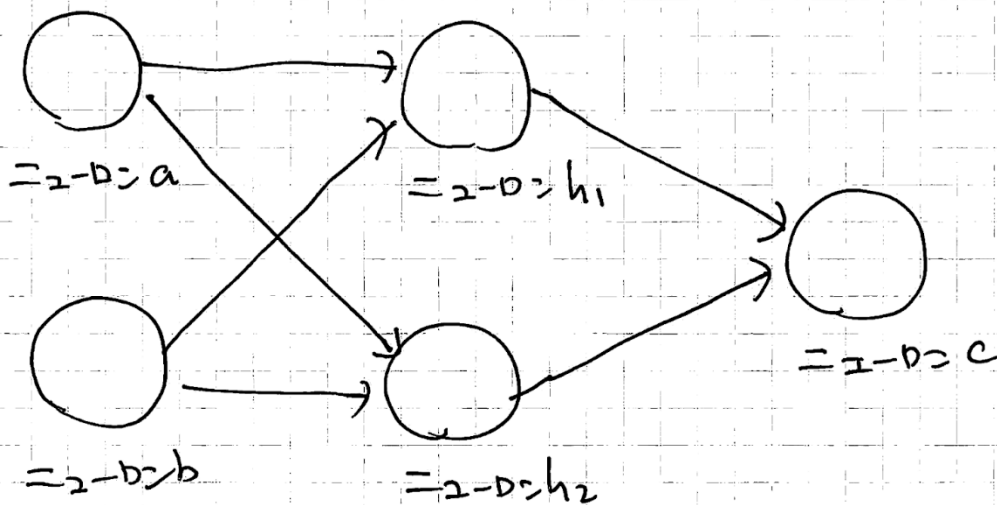
パーセプトロンの能力

多層化することで既知のニューラルネットワークの機能が組み合わされて新しい機能が生まれた！！

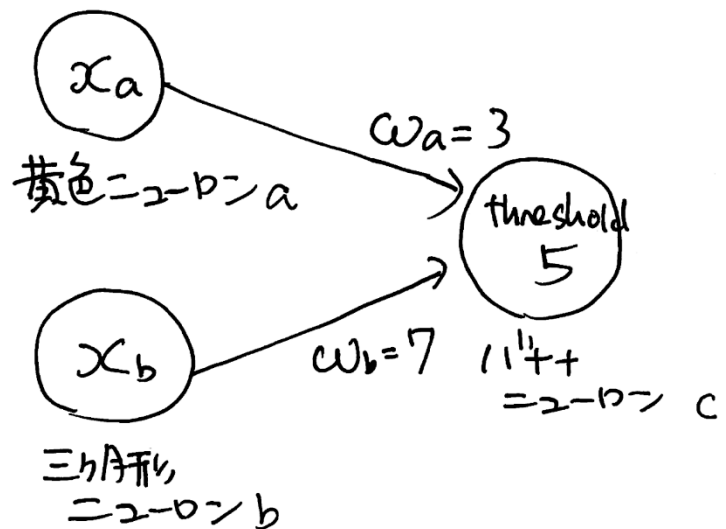
これが深層学習の本質である（すでに深層化された！）

a	b	h ₁	h ₂	c
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

XOR



パーセプトロンの能力



a	b	c
0	0	0
1	0	0
0	1	1
1	1	1

BANANA

バナナ判別機（あくまでも概念です）

$$y_c = H(3x_a + 7x_b - 5)$$

ニューロンBに荷重が偏重しているモデル
（こうした偏重した荷重を学習していくのが深層学習である）

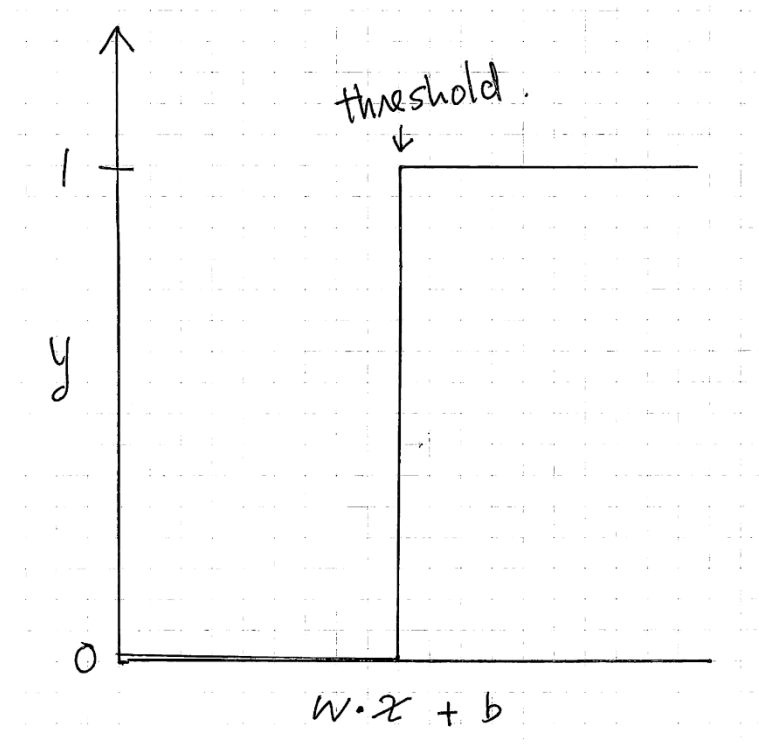
現代の人工ニューロン

$$y=f(\mathbf{w} \cdot \mathbf{x} + b)$$

- 1) threshold をバイアス項bに変更
- 2) 階段方程式Hを活性化関数 f に変更
- 3) 活性化関数 f は様々な非線形関数であり、表現力が各段に上昇
- 4) 深層学習で重要な数式

各種活性化関数

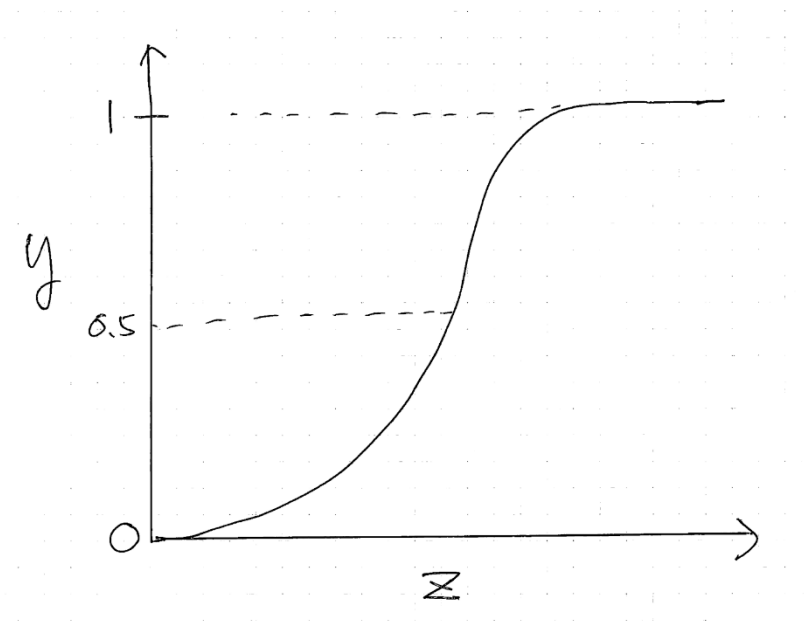
$$y = f(\mathbf{w} \cdot \mathbf{x} + b)$$



階段方程式

各種活性化関数

$$y = f(\mathbf{w} \cdot \mathbf{x} + b)$$



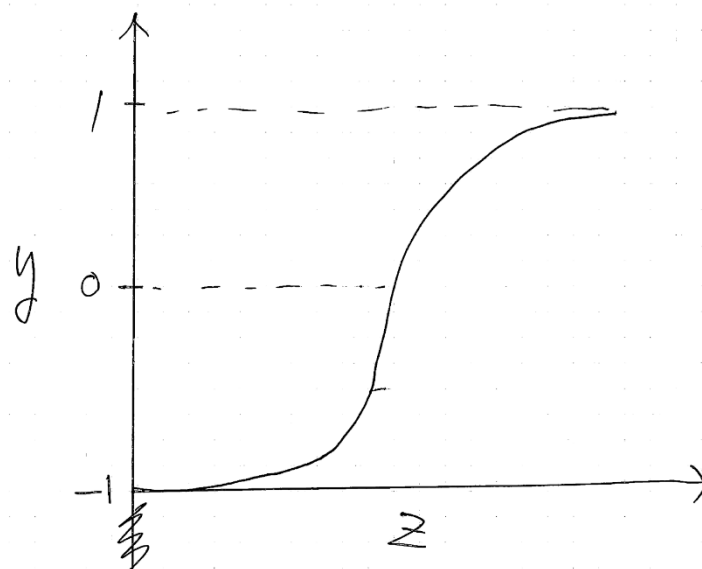
$$f(z) = \frac{1}{1 + e^{-z}}$$

$$(z = \mathbf{w} \cdot \mathbf{x} + b)$$

シグモイド

各種活性化関数

$$y = f(\mathbf{w} \cdot \mathbf{x} + b)$$



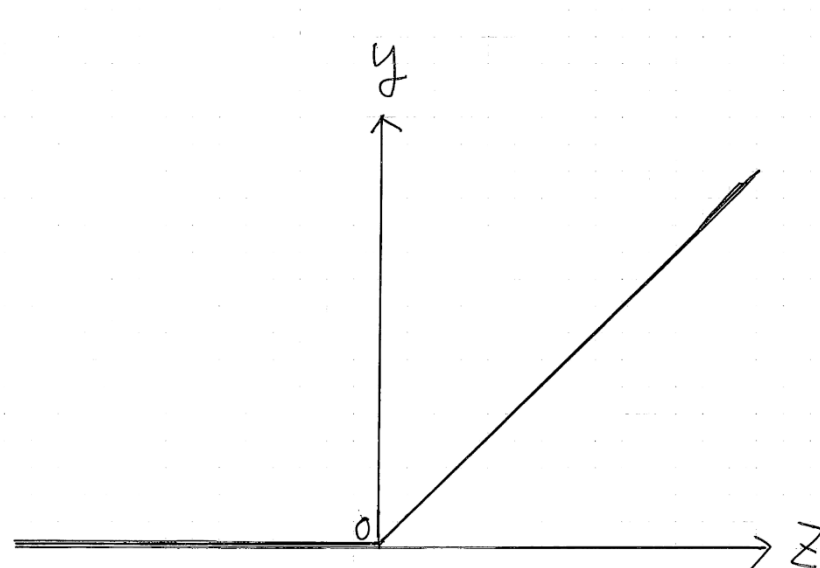
$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$(z = \mathbf{w} \cdot \mathbf{x} + b)$$

Tanh (ハイパーボリックタンジェント)

各種活性化関数

$$y = f(\mathbf{w} \cdot \mathbf{x} + b)$$



$$f(z) = \max(0, z)$$

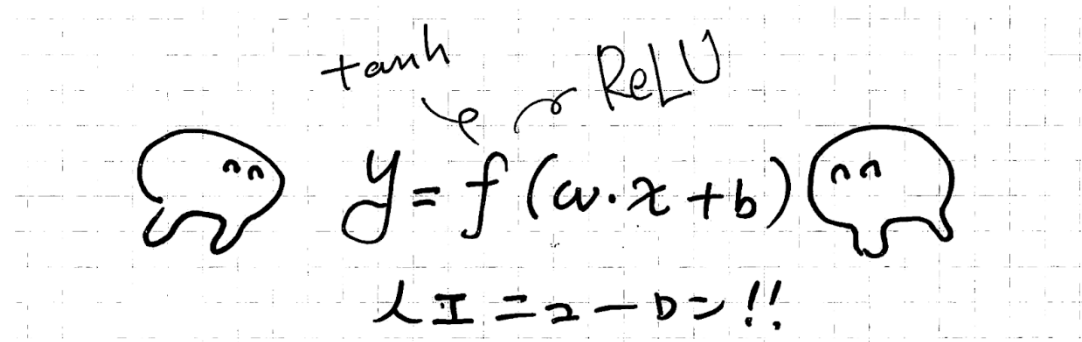
$$(z = \mathbf{w} \cdot \mathbf{x} + b)$$

正規化線形ユニット (rectified linear unit、ReLU)

AlexNetで採用された画期的な活性化関数

(2012年の物体の認識率を競うILSVRCにおける、
GPU利用による大規模深層学習(ジェフリー・ヒントン率いる研究チーム))

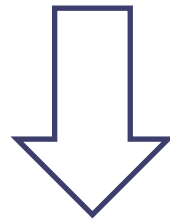
ニューラルネットワーク関数



A hand-drawn diagram on a grid background. It features a central equation $y = f(w \cdot x + b)$ flanked by two speech bubble icons. Above the equation, the word "tanh" is written with an arrow pointing to the function f , and "ReLU" is written with an arrow pointing to the same function. Below the equation, the text "人工ニューロン!!" (Artificial Neuron!!) is written.

$$y = f(w \cdot x + b)$$

人工ニューロン!!



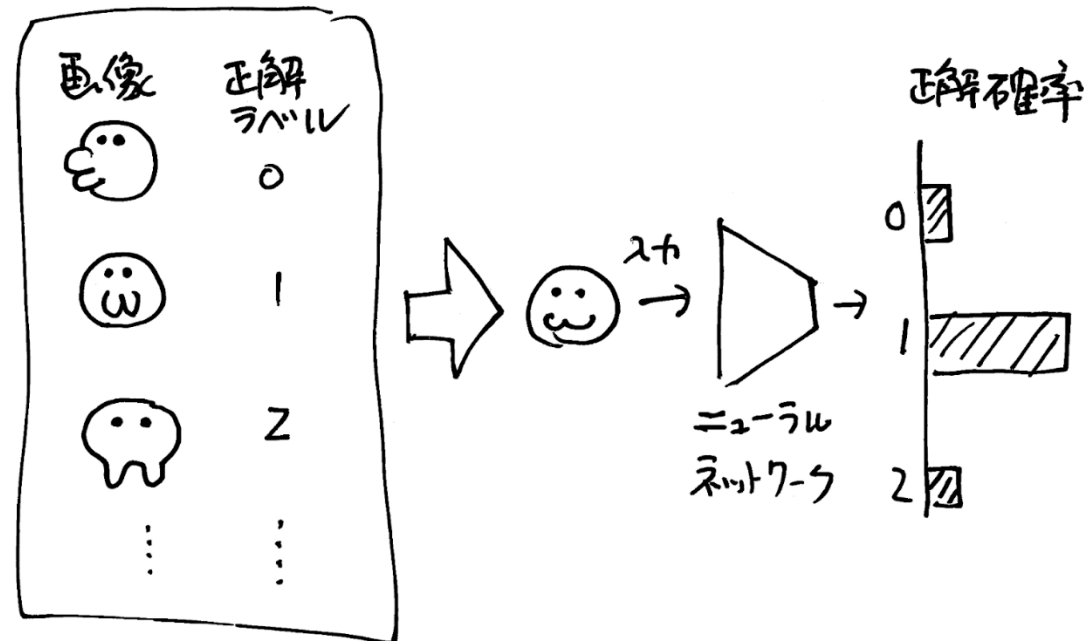
$$y = \text{NeuralNet}(x; w, b)$$

xが入力であり、w, bがパラメータ、yが出力である。通常、関数中の引数を分ける場合はカンマ(,)を使うが、ここではwとbというパラメータが学習過程で更新される変数であることを明確化するためにセミコロン(;)で、入力と区別をしている。yとxが適切な関係になるようなwとbを探索するのが深層学習

教師あり学習

学習には大きく分けて3種類ある。教師あり学習 (supervised learning)、教師なし学習 (unsupervised learning)、強化学習 (reinforced learning) である。

教師あり学習では、入力 x に対して正解 y が用意されている。入出力 (x, y) に対応する正解 (x, y') である。このような正解 (x, y') の集まりを訓練データ (training data) と呼ぶ。良質な訓練データは非常に重要である。

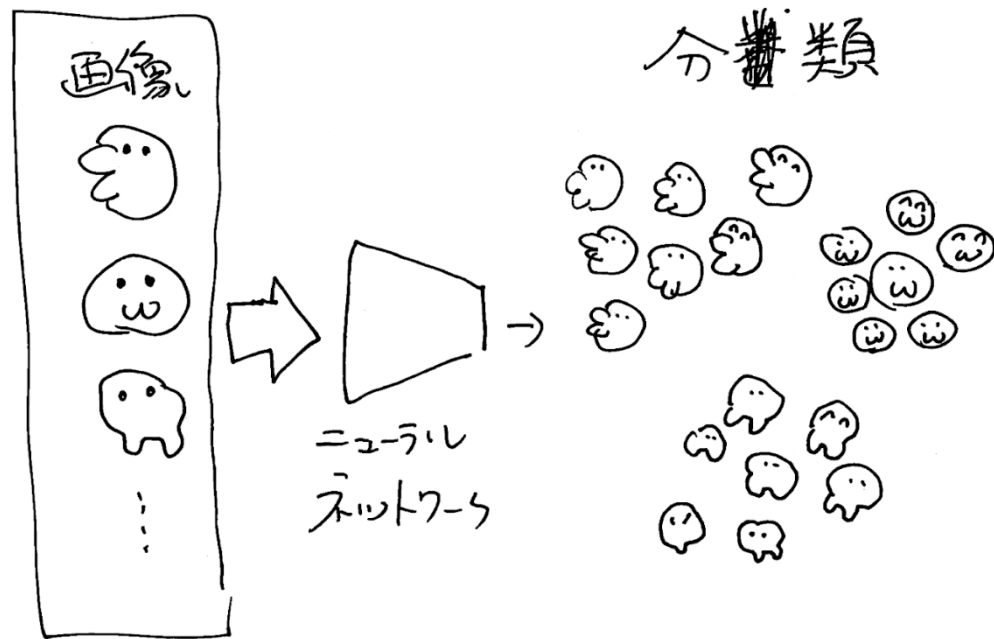


教師あり学習

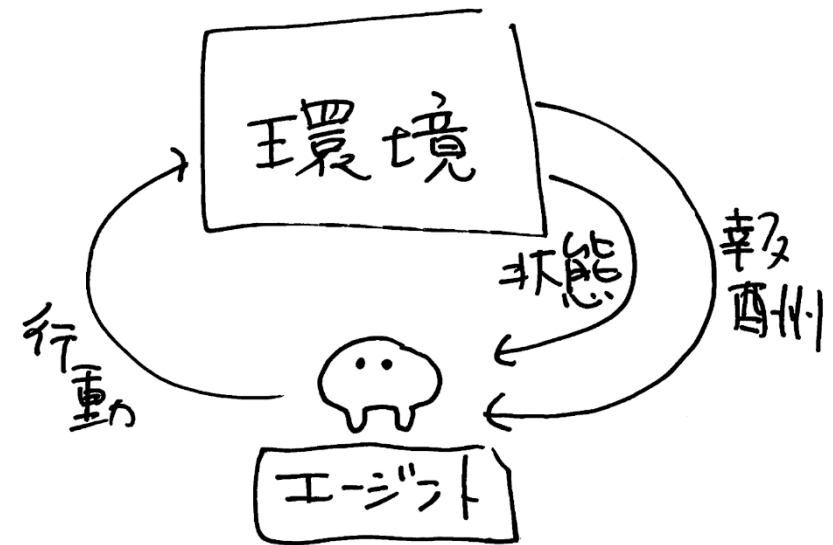
AlexNetという画像分類のニューラルネットワークが画期的な性能を出したと言ったが、これは数々の偶然が結晶化したものである。そのひとつに訓練データの充実が挙げられる。Fei-Fei Leeらが作成した大規模な公開データセットであるImageNetの貢献は絶大であった（名前がややこしいので注意。Netと最後についているが、ImageNetはニューラルネットワークではない。データセットの名前である）。ImageNetには数千万枚の画像が備わっており、数万件のカテゴリーに分類されている。AlexNetは、ImageNetのデータセットを利用して開催された画像分類のコンテスト（ILSVRC、ImageNet Large Scale Visual Recognition Challenge）の3年目2012年に現れ、圧倒的な性能で優勝をした。それが今の深層学習ブームのスタート地点である。多くの人はAlexNetそのものに目を惹かれるが、AlexNetの性能を作り上げた大きな要因のひとつに優れた訓練データにあったのである。



教師なし学習と強化学習



教師なし学習



強化学習

損失関数

$$y = \text{NeuralNet}(x; w, b)$$

$$y \text{ vs } y'$$

NeuralNetが推定値として出力する y に対して、正解値 y' との差が誤差である。この誤差を最小化する方向へ学習を進めることになる。そのときに誤差を計算する方法が損失関数である。

損失関数

平均二乗誤差

平均二乗誤差は、予測値と実際の値のズレを二乗して平均して算出する。深層学習など機械学習だけではなく統計の回帰モデルなどで、モデルの精度を確認するためによく使われる手法である。

$$(\text{💡})^2 = \text{💡💡}$$

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

ここで

y_i は 正解値

\hat{y}_i は 予測値

n は 正解値の 数

損失関数

クロスエントロピー誤差

クロスエントロピー損失（cross-entropy cost）は、分類問題において最も一般的に使用される損失関数の一つである。特に、ソフトマックス関数と組み合わせて使用されることが多い。クロスエントロピー損失は、モデルの予測確率分布と実際の確率分布（正解ラベルの分布）との差を測る指標として機能する。1対1の数字の比較ではなく、多対多のグループ同士の比較である。

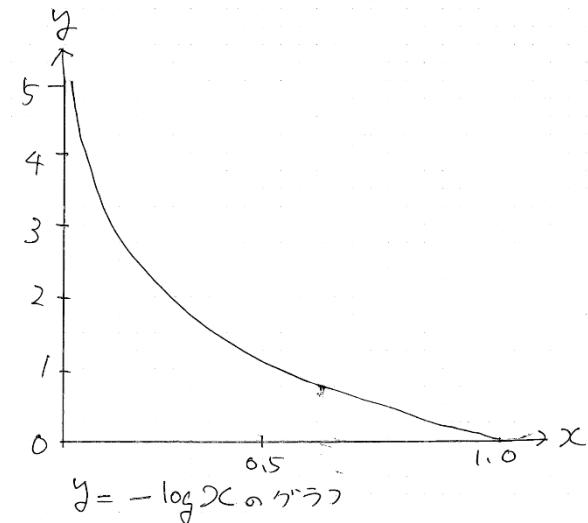
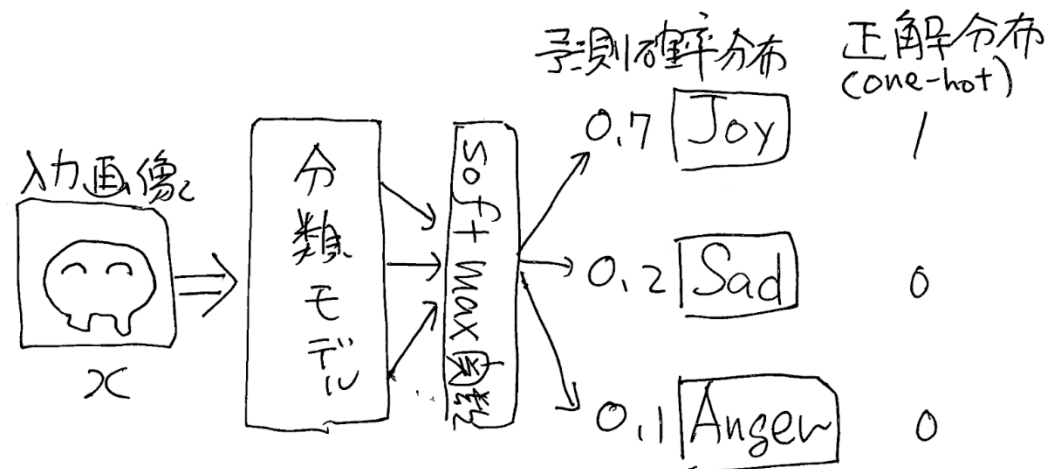
損失関数

図の例では入力画像に対して (0.7, 0.2, 0.1) のようなベクトルで予測確率分布が表現される。これに対して正解の確率分布はone-hotベクトルで表現されている。one-hotベクトルは、正解だけを1にして残りは全て0で表現されているベクトルである。写真の正解がjoyなら (1, 0, 0) というベクトルになる。これが正解ラベルの分布である。二つの確率分布が決まると、以下の数式でクロスエントロピー損失CECが計算できる。

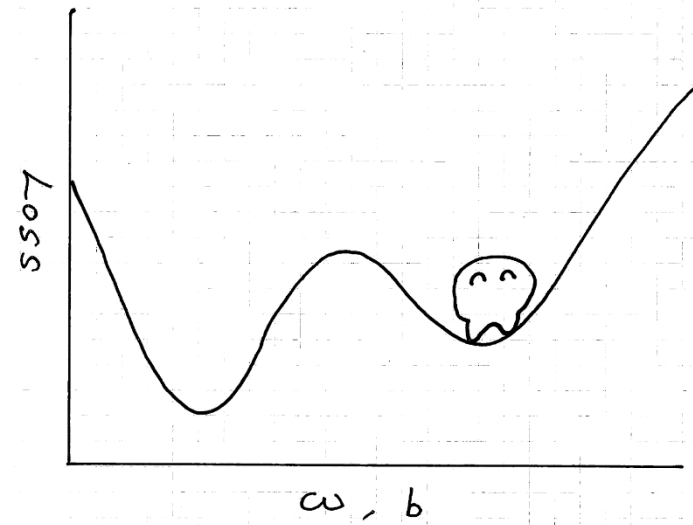
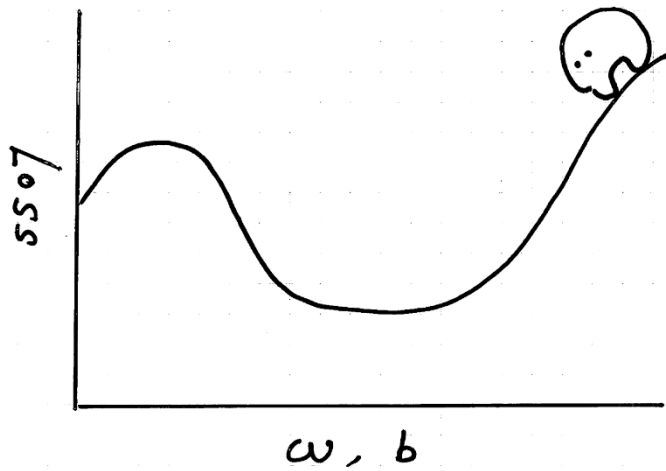
$$\text{CEC} = -\sum(y_i' * \log(y_i))$$

ここで、 y_i' は正解ラベル (0または1)、 y_i はモデルの予測確率、 i はクラスの指標である。正解は一つだけなので、正解クラスのみから数値が出てくる。なので正解クラスの予測確率を y とすると、CECは以下のように計算される。

$$\text{CEC} = -\log(y) = -\log(0.7) = 0.36$$



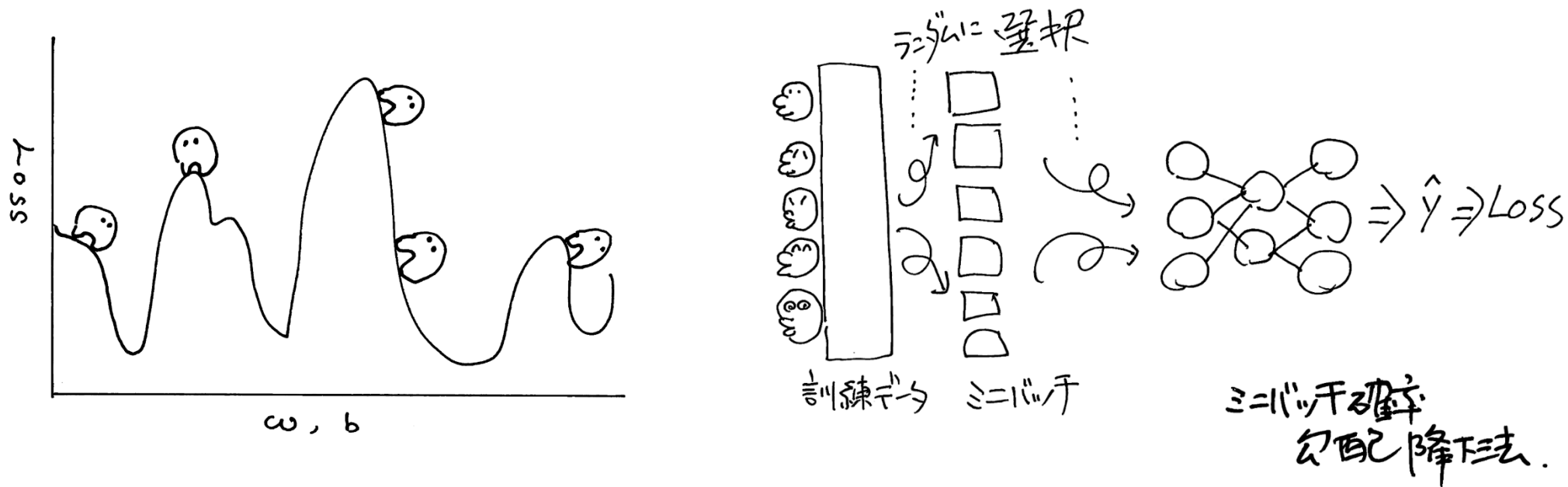
誤差逆伝播法



局所の勾配（微分値）に沿ってパラメータを変更する（勾配降下法）

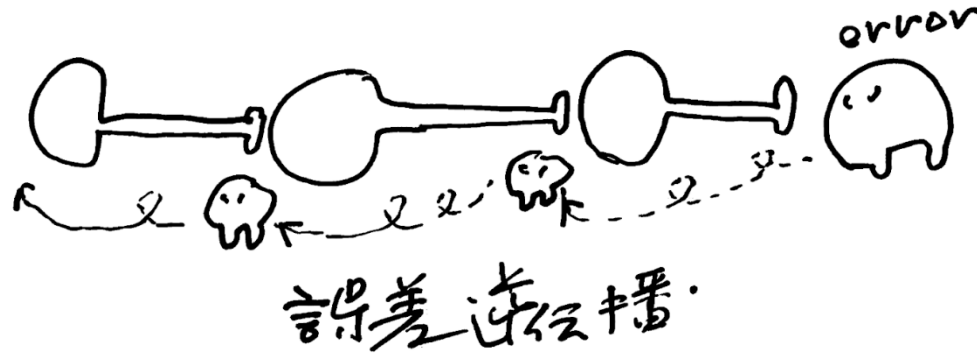
活性化関数は微分可能であることが必要！

誤差逆伝播法



局所解から逃れる方法が色々考案されている

誤差逆伝播法



ひとつひとつのニューロンに対してランダムに損失関数の勾配を計算していると非常に効率が悪い。そこで考え出されたのが誤差逆伝播法である。誤差逆伝播法では出力層における損失値からニューラルネットワークを逆方向に1層ずつ入力方向へ戻していき、各層における各ニューロンの勾配を計算していく。戻していく際に、先ほど説明を飛ばした偏微分の算術を巧みに駆使する。偏微分には連鎖の法則（chain rule）という計算方法があり、繋がっているニューロンであれば誤差を伝播させていくことができ一気に計算することができる（大発明である）。誤差逆伝播法を行うためにも活性化関数の微分可能性は重要。

誤差逆伝播法は、一般的にはHintonらのグループの発明とされているが、実は多くのグループが時代を超えて似たようなアイデアを再発明している。日本の甘利俊一氏もそのひとりで、Hintonらのグループよりもかなり早く提案している。ただそれを実行するコンピュータがなかったため注目度は低かった。時代の縁というしかない。

$$y = \text{NeuralNet}(x; w, b)$$

ニューラルネットワークは人工ニューロンの塊で出来上がった関数であり、学習の種類と目的関数と訓練データが決めれば、あとは学習によって適切なパラメータを獲得すればモデルになる

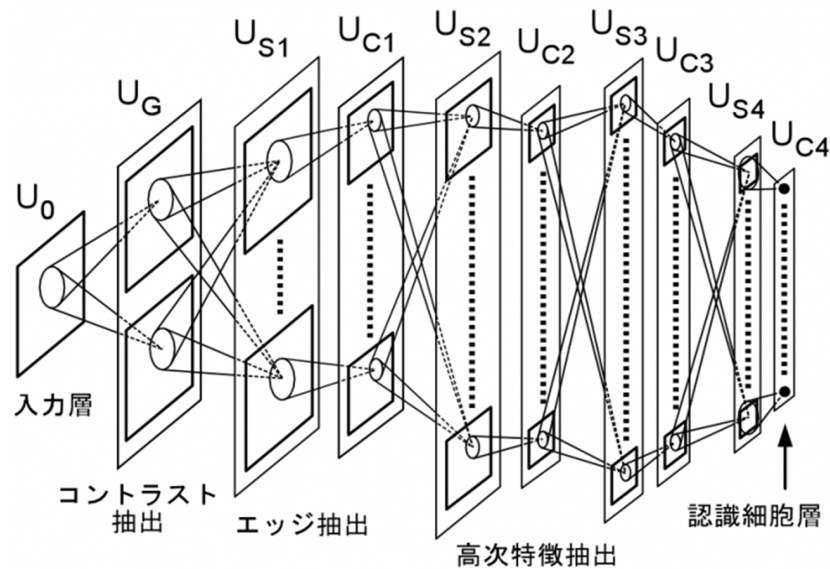
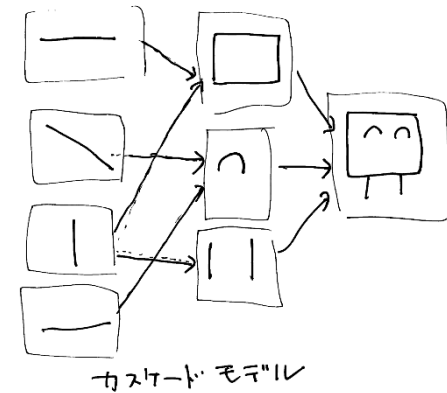
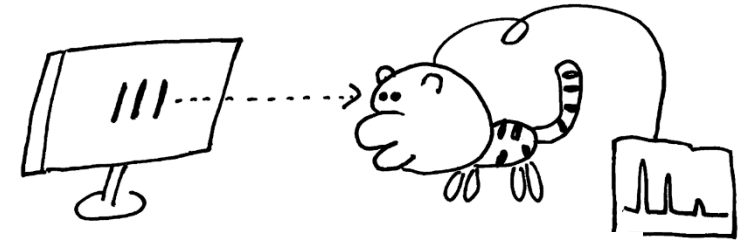
畳み込みニューラルネットワーク

David HubelとTorsten Wieselの画期的な発見（1959）

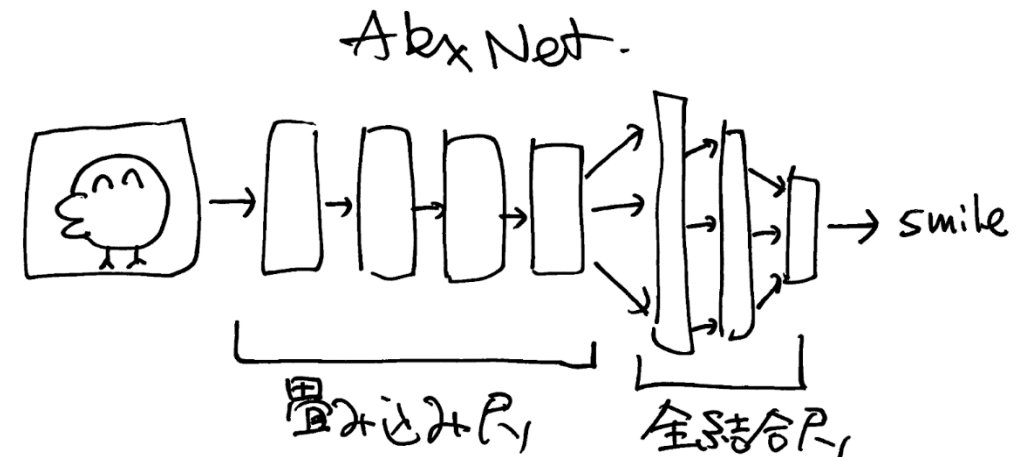
福島邦彦のネオコグニトロン（1979）

Yann LeCunとYoshua BengioによるLeNet-5（1998）

Geoffrey HintonらのグループによるAlexNet（2012）



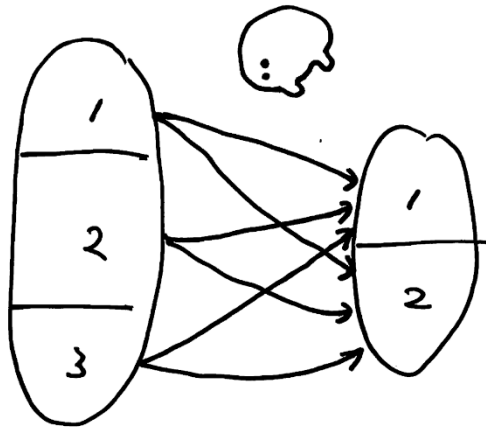
Neocognitron ©福島邦彦



畳み込みニューラルネットワーク

全結合層

全結合層は、結合を送り出しているプレニューロンのある層と、結合を受け取るポストニューロンのある層を合わせた構造である。通常の深層学習では結合は層から層へと一方通行なので、全結合層のプレニューロン数を N 、ポストニューロン数を M とすると荷重パラメータ数は $N \times M$ になり、ニューロン数あたりでは最も大きなパラメータ数を持つ層構造である。パラメータ数が大きいと自由度が高く、それだけモデルの表現力は豊かになるので、全結合層は最も豊かな表現が可能である。後に紹介するすべての結合層は原理的には全結合層で表現できる。

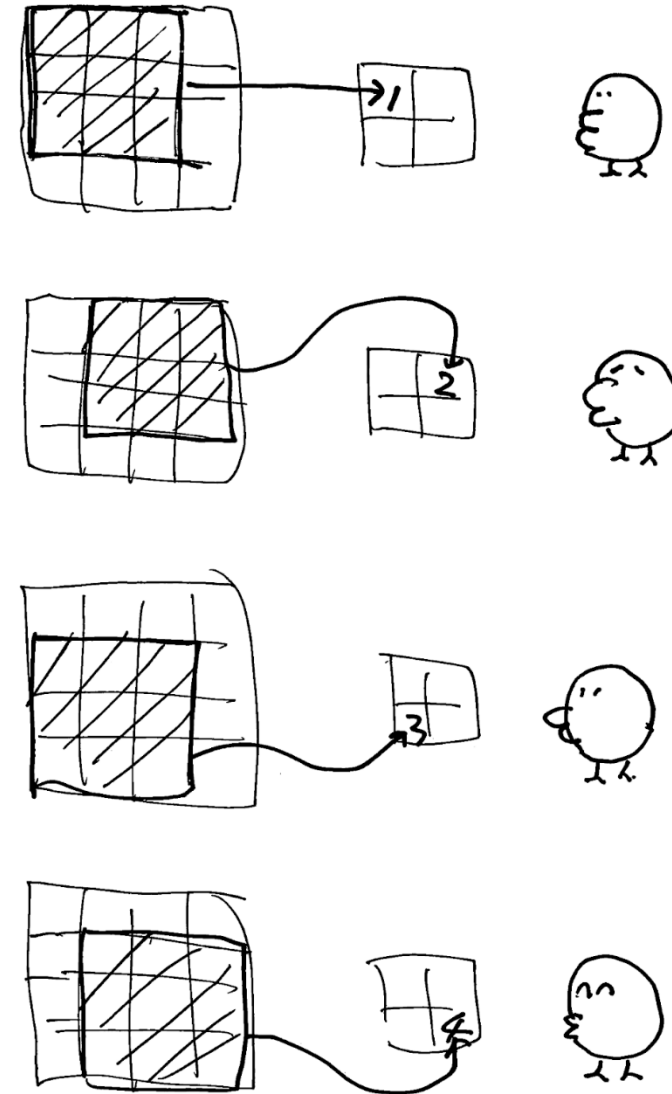


畳み込みニューラルネットワーク

畳み込み層

畳み込み層は全結合層と異なりプレニューロンは近い位置にあるポストニューロンだけと結合する。そのため全結合層と比べると結合数を大幅に減らすことが出来る。しかも荷重パラメータはカーネルと呼ばれるフィルターにパッケージングされているために、さらにパラメータ数は大幅に減らすことができる。その原理を図に示した。

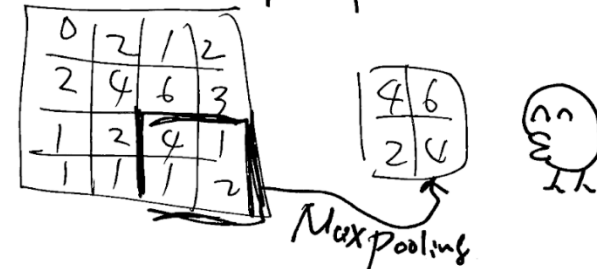
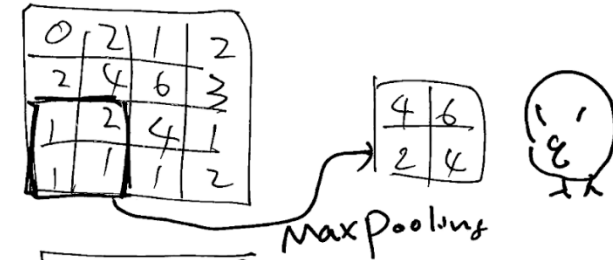
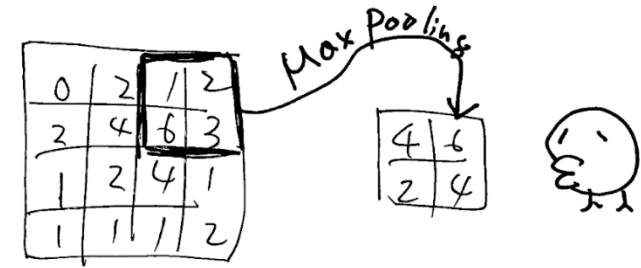
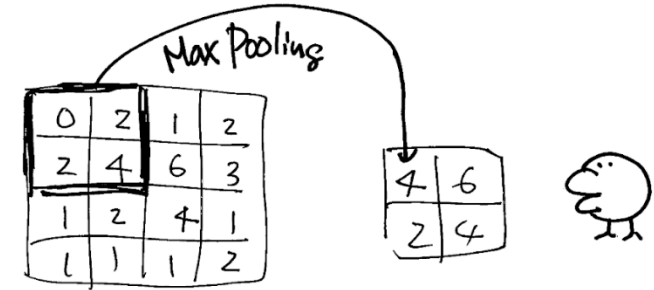
カーネルは「横幅3 x 縦幅3」や「横幅5 x 縦幅5」といった正方形が一般的である。例えば3 x 3カーネルの場合、9個のプレニューロンが一つのポストニューロンへ結合するための荷重パラメータがパッケージングされている。隣り合う9個のニューロンの関係性、あるいは9個のニューロンが表現している特徴がカーネルによって抽出される。



畳み込みニューラルネットワーク

プーリング層

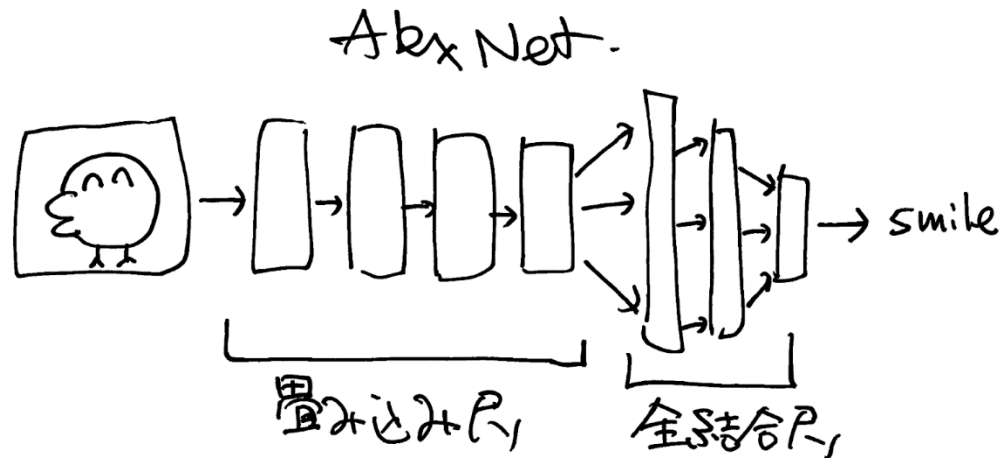
プーリング層にもカーネル処理が用いられる。その意味では畳み込み層の一種である。ただしプーリング層のカーネルは、近傍のニューロンの活動の平均値 (Averaged Pooling) や最大値 (Max Pooling) を出力するだけで、学習すべき荷重パラメータを持たない。とてもユニークなニューラルネットワークである。近傍ニューロンの平均値や最大値を採用することでノイズの除去が可能となる。



畳み込みニューラルネットワーク

畳み込み層！
プーリング層！
全結合層！
これでAlexNetが作れる！

GoogleColabで実装してみましょう。
(alexnet.ipynbをGPU環境で実行)



AlexNetクラス

```
class AlexNet(nn.Module):

    def __init__(self, num_classes):
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Linear(256 * 4 * 4, 4096),
            nn.ReLU(),
            nn.Dropout(),
            nn.Linear(4096, 4096),
            nn.ReLU(),
            nn.Linear(4096, num_classes),
        )

    def forward(self, x):
        x = self.features(x)
        x = x.view(x.size(0), 256 * 4 * 4)
        x = self.classifier(x)
        return x
```

これで現在のAIブームの起爆剤となったAlexNetが理解できました。その後沢山の深層ニューラルネットワークが作られましたが、すべて人工ニューロンがコアにあります。

今回で興味を持たれた方は、ご自分でどんどん勉強を進めるなり、「生物データ解析のためのPython AIプログラミングトレーニングコース」をご受講くださるのも一興です。

お疲れ様でした！

生物データ解析のための

Python AIプログラミング トレーニングコース

受講料
無料

コース日程

2024年

9月18日水 ▶ 19日木

募集開始

2024年7月22日月 ▶ 8月18日日

場所

基礎生物学研究所

講師・世話人

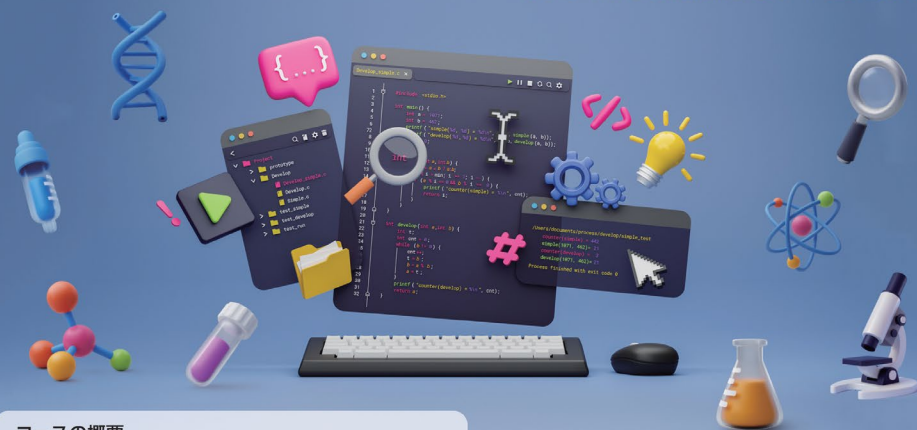
代表: 小山 宏史 (NIBB)

講師

加藤 輝 (NIBB バイオイメージング解析室)・
渡辺 英治 (NIBB AI解析室)・杉浦 宏樹・中村 貴宣・
西出 浩世 (NIBB データ統合解析室)

スーパーバイザ

藤森 俊彦 (NIBB 超階層生物学センター)・上野 直人 (NIBB)



コースの概要

- 生物データ解析のためのpythonプログラミング基礎
- 画像系AIの導入・プログラミング・画像データ解析

主催

自然科学研究機構 基礎生物学研究所 (NIBB) AI解析室
先端バイオイメージング支援プラットフォーム (ABIS)

詳細・受講申込 <https://bioimageanalysis.jp/paitc/>

お問い合わせ 自然科学研究機構 基礎生物学研究所 担当: 杉浦

E-mail sugiura@nibb.ac.jp

TEL 0564-55-7626

