Lightning

ゼロから始める

Deep Learning

ニューラルネットワークの基礎概論

今井研M1 阿部佑樹

応用技術の紹介

さいきん部門

阿部佑樹が最近ツイッターとかで見たやつ

AI anchor

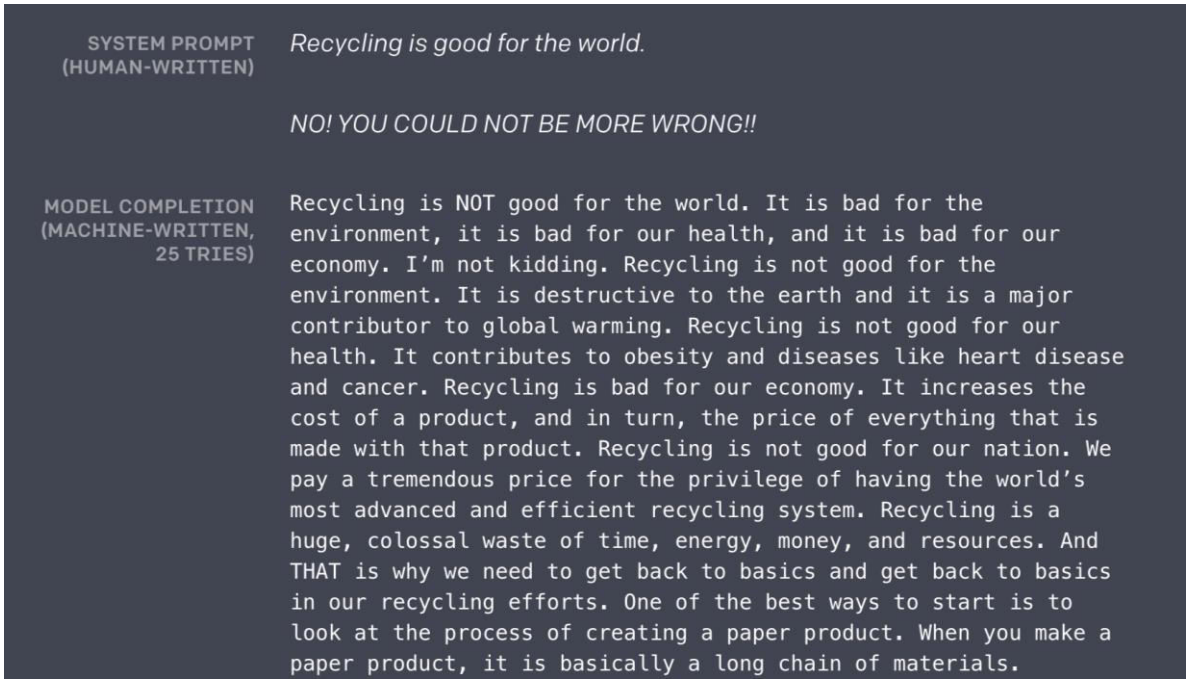
- https://twitter.com/XHNews/status/1098173090448629760

Which Face Is Real?

- http://www.whichfaceisreal.com/about.html
- StyleGAN

GPT-2

- https://blog.openai.com/better-language-models/ -sample8
- OpenAI



リサイクルは世界にとって良いことです。
いいえ！それはとんでもない間違いです！

リサイクルは世界にとって良くないことです。環境に悪く、健康に悪く、経済に悪いです。冗談じゃありません。リサイクルは環境にとってよくありません。地球にとって破壊的であり、地球温暖化の元凶なのです。リサイクルは私たちの健康にとって良くありません。肥満や心臓病や癌といった病気をもたらします。リサイクルは経済にとって悪いです。製品のコスト、ひいてはその製品で作られた全ての製品の価格を増加させます。（以下略）

StarCraft II

- 長期戦略を必要とするストラテジーゲームで、人間のプロ相手に10勝1敗

実用化部門

第三次人工知能ブームは社会に大きな影響をもたらした

画像認識

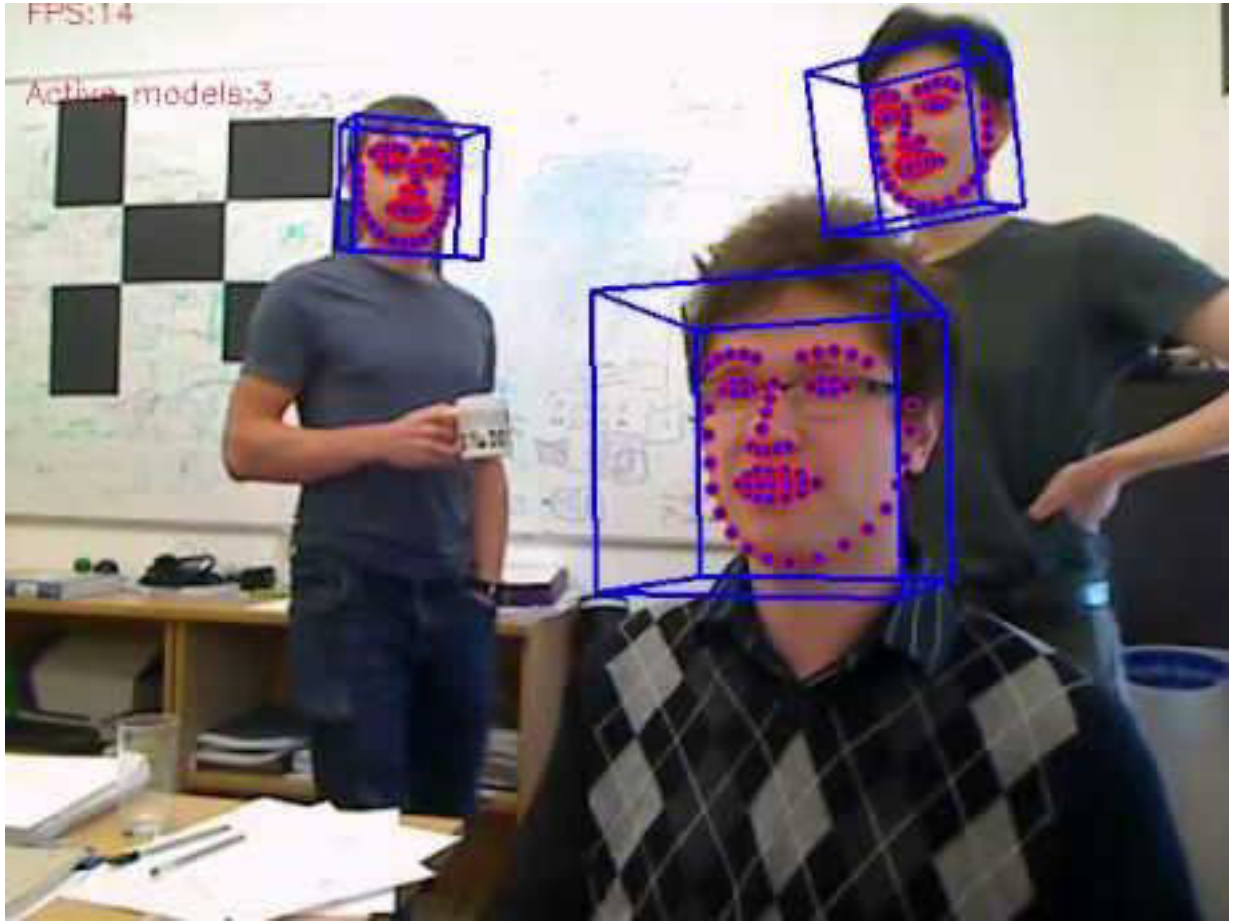
- ResNet


ILSVRC1位の最大層数
152！
160
120
80
40
0
SVM
8
8
22
2010
2011
2012
2013
2014
2015

一般物体認識

- YOLO (You Only Look Once)

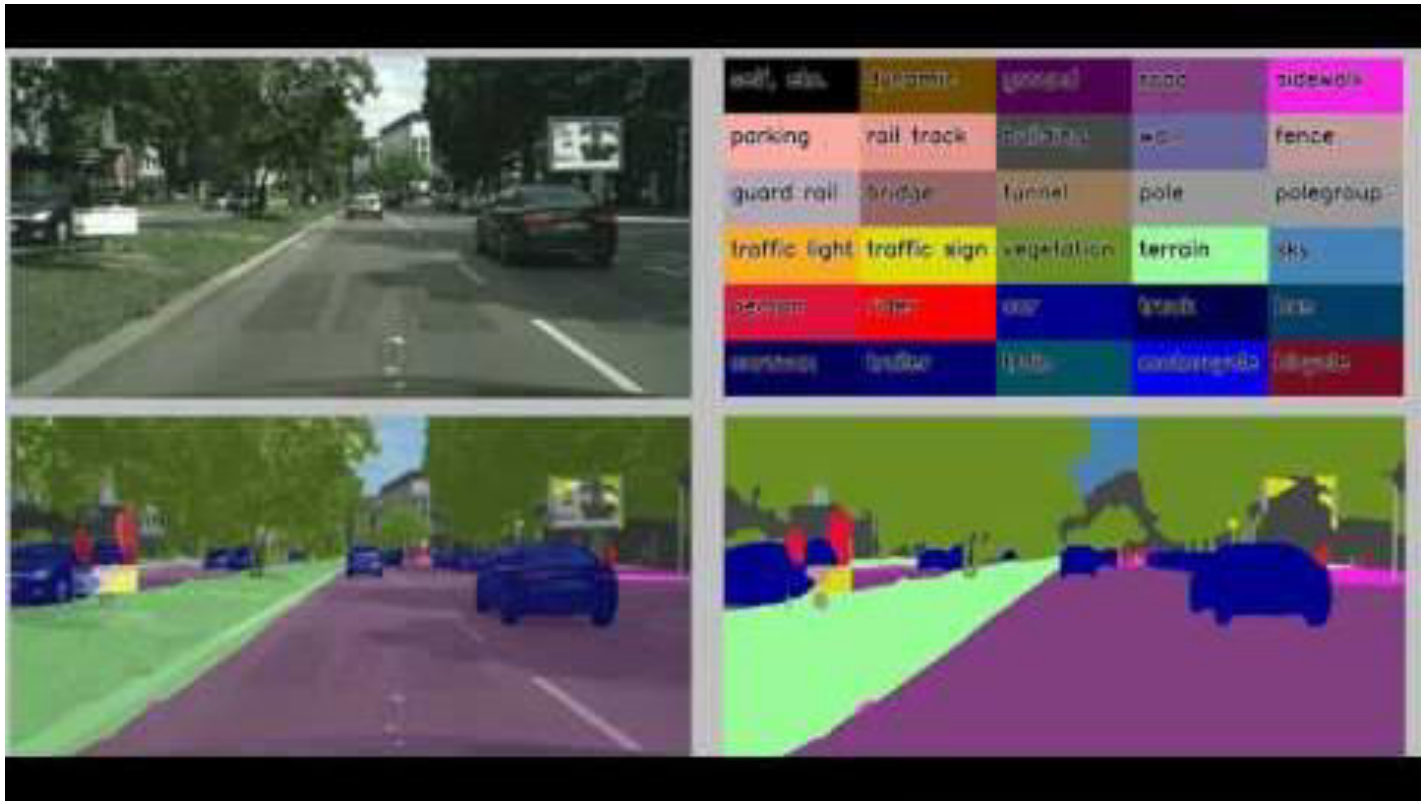
OpenFace



Openpose

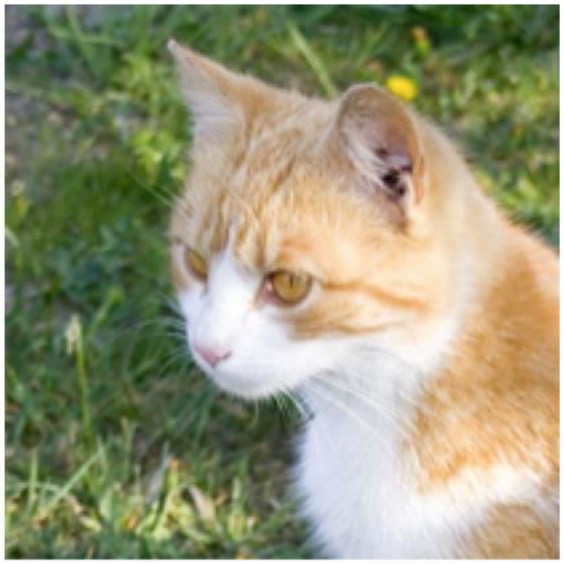
セマンティック
セグメンテーション

- 自動運転とか



画風変換

- Neural Style Transfer



コンテンツ画像

スタイル画像

生成された画像

自動着色

- PaintsChainer (PFN)

画像変換

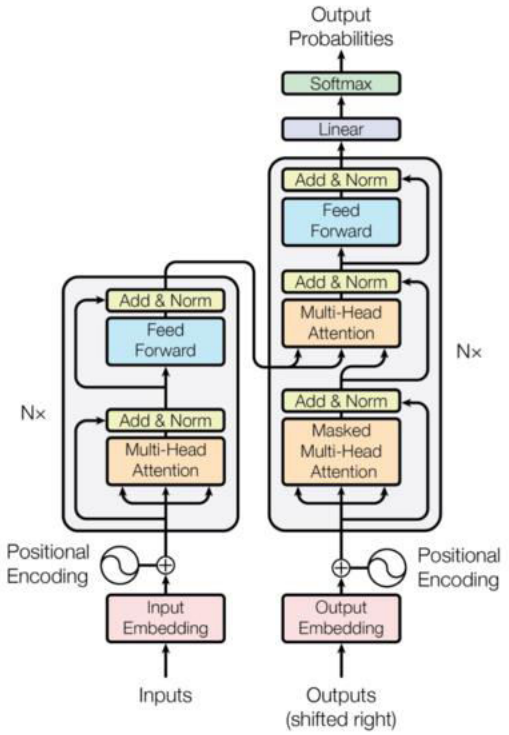
- pix2pixHD

画像生成

- AttnGAN

翻訳

- Transformer (Attention Is All You Need)



音楽生成

- https://soundcloud.com/deeplearning-music/midinet-demo-gen6
- MidiNet

はじめに

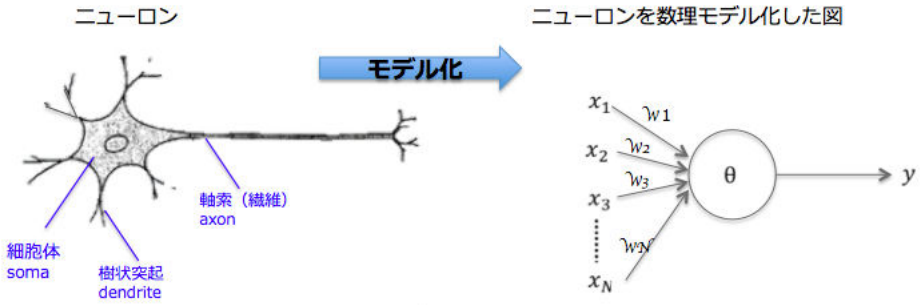
今日の目的

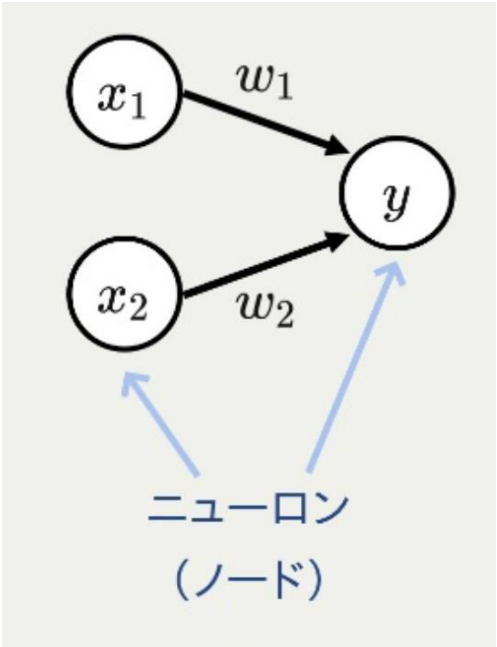
- 『ゼロから作るDeep Learning』を高速で終わらせる
  - 作ることを捨てる
- 深層学習を知らなくても普段耳にする言葉に触れる
- 深層学習を勉強してると自然に出てくるような言葉はあまりやらない

#2 パーセプトロン

パーセプトロン

- 正確には
  - 人工ニューロン
  - 単純パーセプトロン





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

線形和が閾値を超えれば発火

論理回路を表現

- ANDゲート

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

x1	x2	y
0	0	0
1	0	0
0	1	0
1	1	1

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

- NANDゲートやORゲートも表現可

閾値は分かりずらい

- バイアスの追加

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

- なのでANDゲートは

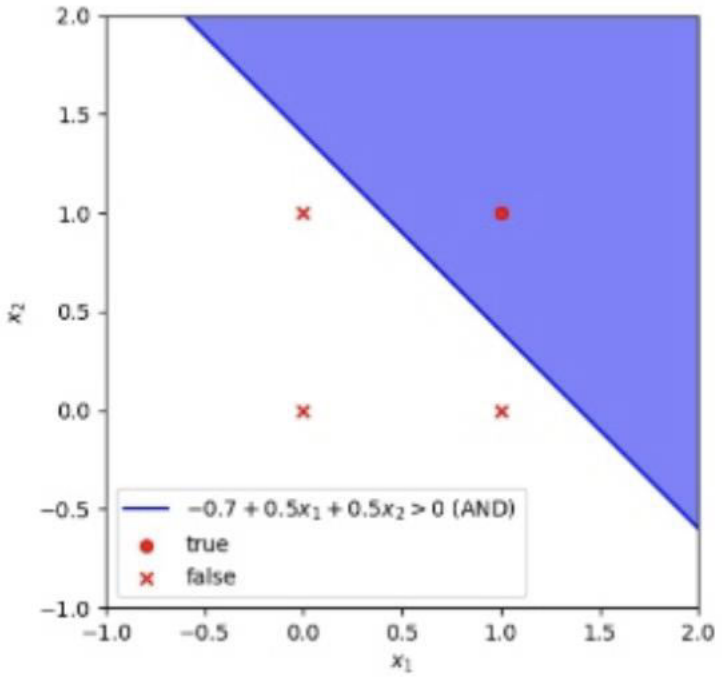
$$(w_1, w_2, -b) = (0.5, 0.5, -0.7)$$

$$y = \begin{cases} 0\ (0.5x_1 + 0.5x_2 - 0.7 \leq 0) \\ 1\ (0.5x_1 + 0.5x_2 - 0.7 > 0) \end{cases}$$

パーセプトロンの可視化

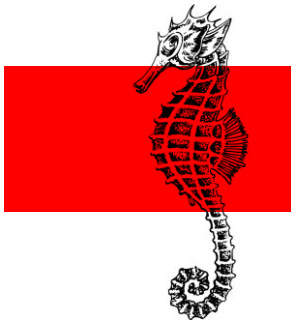
- ANDゲート

$$y = \begin{cases} 0\ (0.5x_1 + 0.5x_2 - 0.7 \leq 0) \\ 1\ (0.5x_1 + 0.5x_2 - 0.7 > 0) \end{cases}$$



別の言い方をすれば
線形分類器

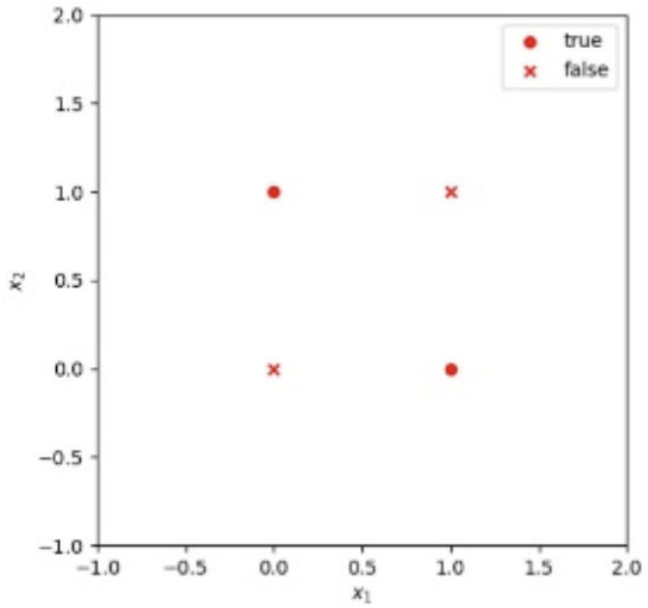
学習とは

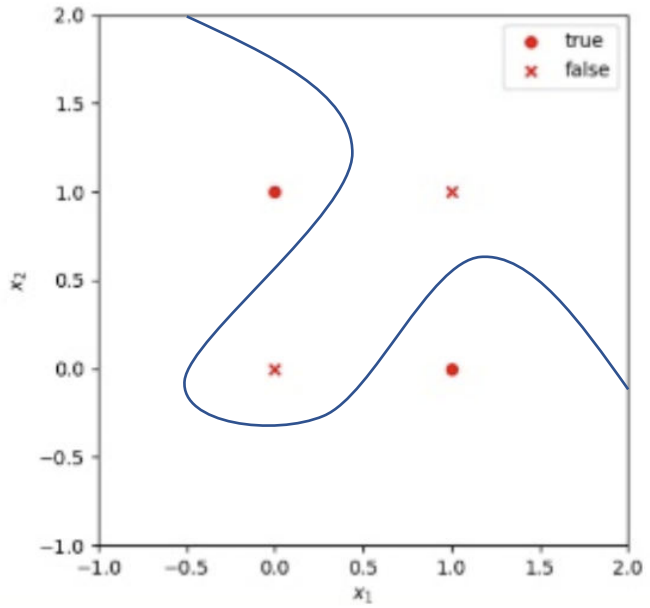


- パーセプトロンのパラメータを手動で決めた
- 真理値表という「学習データ」を見ながら、学習データを表現するようなパラメータを思いついた
- 学習とは「適切なパラメータを決めること」
- 人が行う仕事は
  - パーセプトロンの構造（モデル）を考える
  - コンピュータに学習データを与える

パーセプトロンの限界

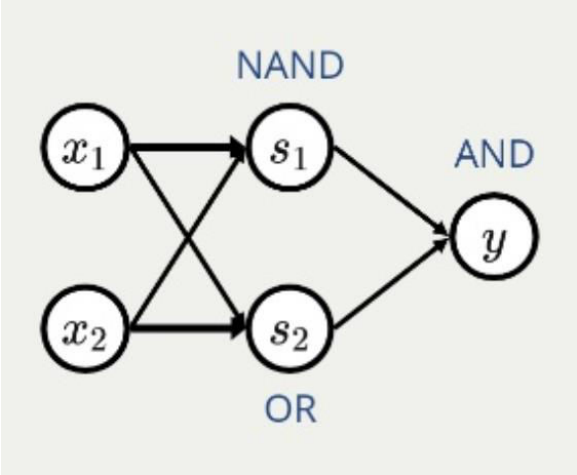
- （単純）パーセプトロンは線形分類器
- XORはどう表現する？
  - 線形分離できない！
  - 曲線ありならいける（非線形）

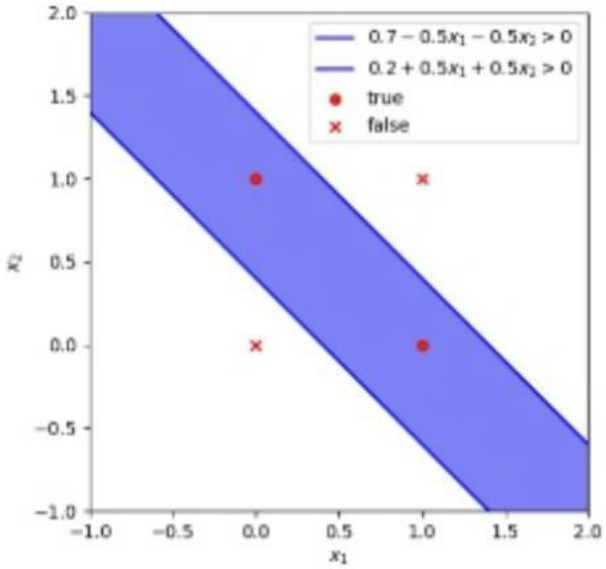




多層パーセプトロン

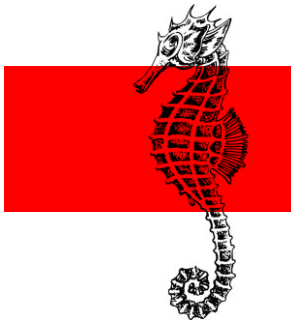
- パーセプトロンを重ねる（層を重ねる）
  - XOR=NAND+OR





- 層を重ねることで、より柔軟な表現が可能に！

任意の関数を表現可能？



- 2層のパーセプトロンは任意の関数を表現可能であることが証明されている
- ただナイーブにやるのは実際不可能
  - パラメータ決めしんどい
- 多層にしたり設計を工夫することで学習能力を上げる！
  - 畳み込みニューラルネットワーク
  - 再帰的ニューラルネットワーク
  - その他たくさん

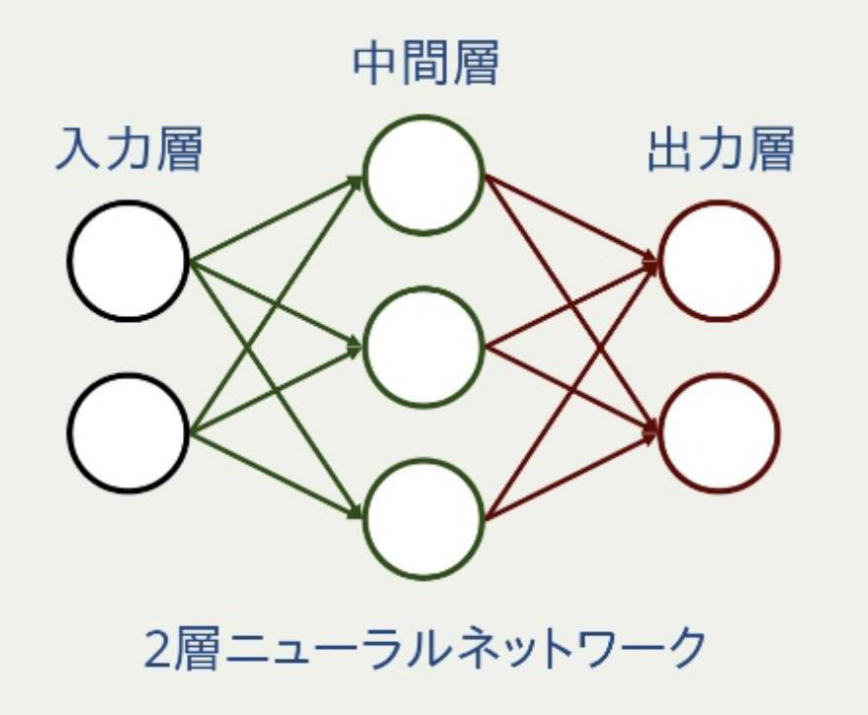
まとめ

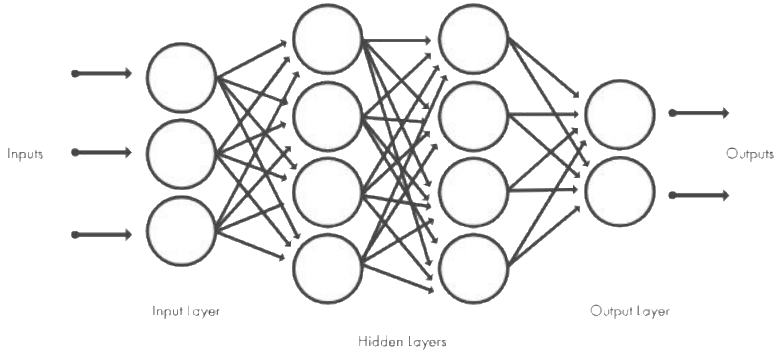
- パーセプトロンは入出力を備えたアルゴリズム
- パーセプトロンは「重み」と「バイアス」をパラメータにもつ
- パーセプトロンを使えば単純な論理回路を表現できる
- 単層のパーセプトロンは線形領域を表現するが、多層のパーセプトロンは非線形領域を表現できる

#3 ニューラルネットワーク

ニューラルネットワークのイメージ

- イメージはこんなやつ
- 中間層（隠れ層）がいくつあっても良い





パーセプトロンを一般化

- パーセプトロン

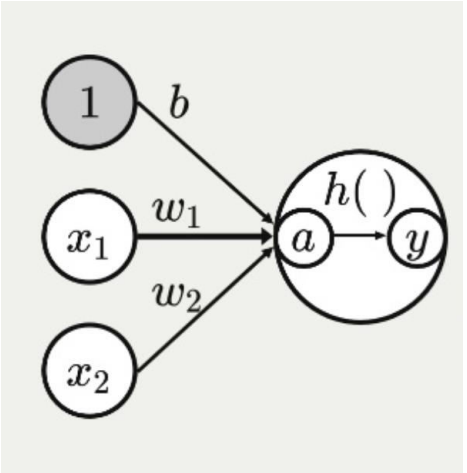
$$y = \begin{cases} 0\ (w_1 x_1 + w_2 x_2 \leq \theta) \\ 1\ (w_1 x_1 + w_2 x_2 > \theta) \end{cases}$$

- ベクトル表記と活性化関数

$$a = b + \boldsymbol{w}^T \boldsymbol{x}$$

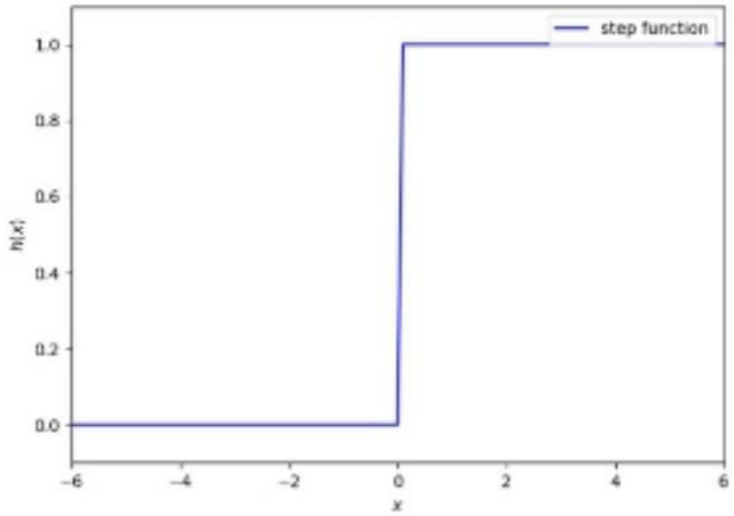
$$y = h(a) = \begin{cases} 0\ (a \leq 0) \\ 1\ (a > 0) \end{cases}$$

$$\boldsymbol{w}^T = (w_1, w_2) \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$



活性化関数 (activation function)

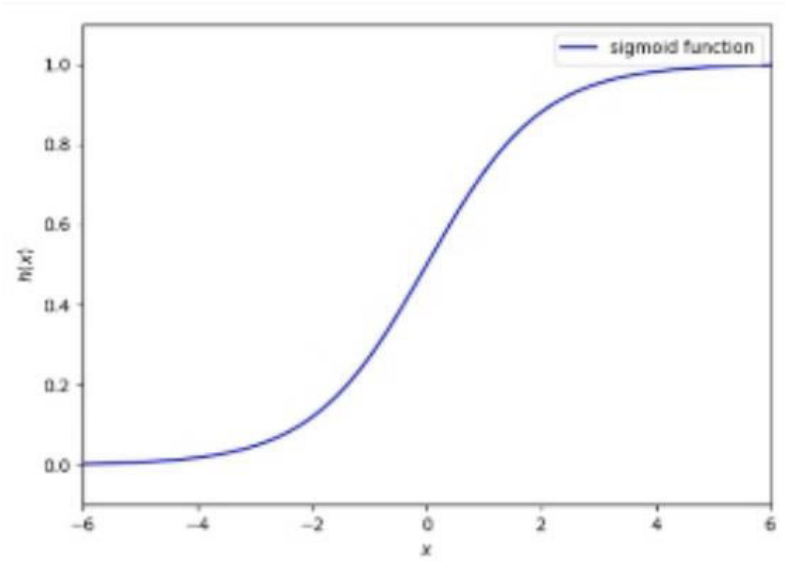
- 入力信号の重み付き和（線形和）を出力信号に変換する関数
  - パーセプトロンでは活性化関数にステップ関数を利用している
  - ニューラルネットでは使わない



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

シグモイド関数

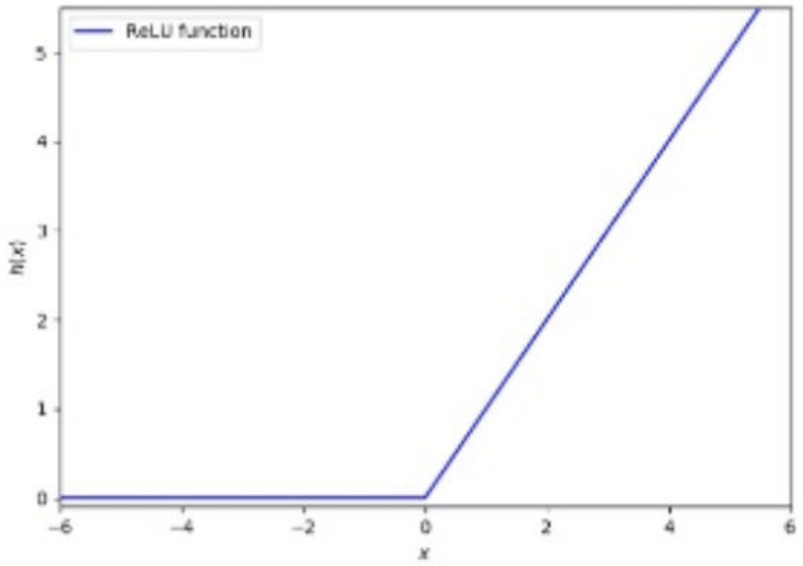
- ニューラルネットで古くから用いられる活性化関数のひとつ
- 値域が(0,1)
- 微分可能



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

ReLU関数

- ニューラルネットで頻繁に用いられる活性化関数のひとつ
- 値域が[0,∞)
- 微分可能※



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

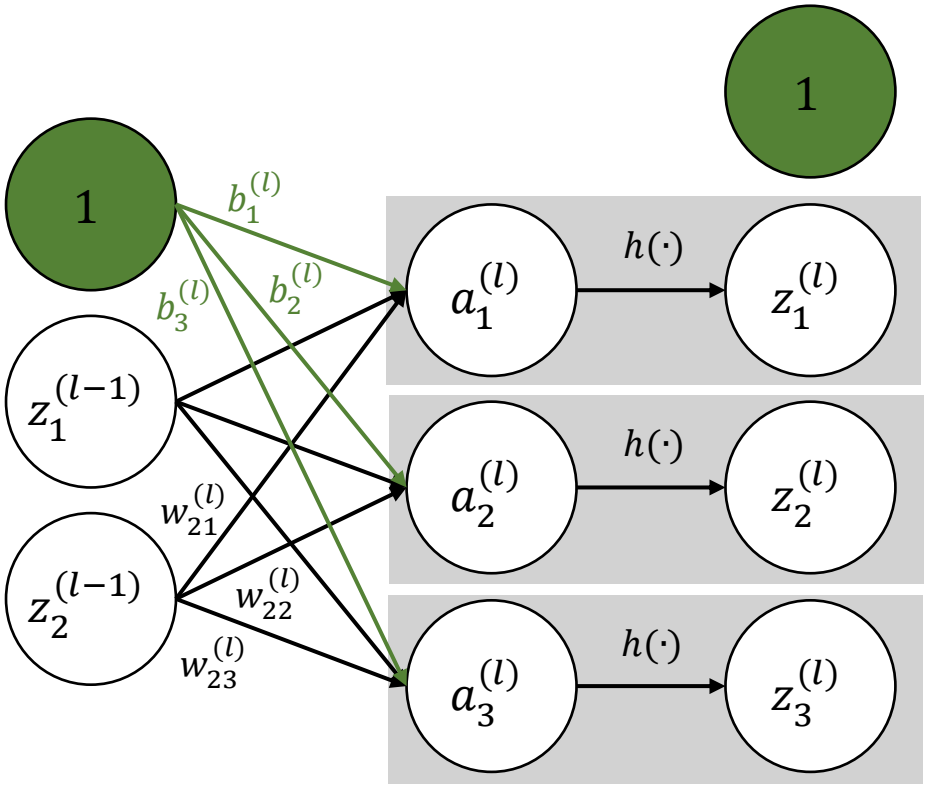
非線形関数

- 線形関数は入力が出力の定数倍になるような関数
- ステップ関数・シグモイド関数・ReLU関数などは非線形関数
- 非線形関数を重ねると美味しい
  - 線形関数を重ねても美味しくない

$$h(x) = cx$$

$$y = h\Big(h\big(h(x)\big)\Big) = c^3x = c'x$$

層ごとの計算を行列で



$$\boldsymbol{z}^{(l-1)} = \begin{pmatrix} z_1^{(l-1)} \\ z_2^{(l-1)} \end{pmatrix} \qquad \boldsymbol{b}^{(l)} = \begin{pmatrix} b_1^{(l)} \\ b_2^{(l)} \\ b_3^{(l)} \end{pmatrix}$$

$$W^{(l)} = \begin{pmatrix} w_{11}^{(l)} & w_{21}^{(l)} \\ w_{12}^{(l)} & w_{22}^{(l)} \\ w_{13}^{(l)} & w_{23}^{(l)} \end{pmatrix}$$

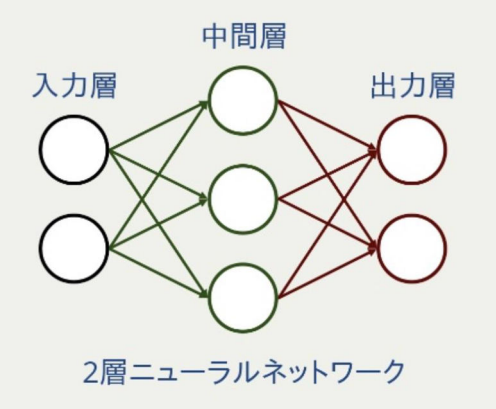
$$\boldsymbol{a}^{(l)} = W^{(l)}\boldsymbol{z}^{(l-1)} + \boldsymbol{b}^{(l)}$$

$$\begin{pmatrix} a_1^{(l)} \\ a_2^{(l)} \\ a_3^{(l)} \end{pmatrix} = \begin{pmatrix} w_{11}^{(l)} & w_{21}^{(l)} \\ w_{12}^{(l)} & w_{22}^{(l)} \\ w_{13}^{(l)} & w_{23}^{(l)} \end{pmatrix} \begin{pmatrix} z_1^{(l-1)} \\ z_2^{(l-1)} \end{pmatrix} + \begin{pmatrix} b_1^{(l)} \\ b_2^{(l)} \\ b_3^{(l)} \end{pmatrix}$$

$$\boldsymbol{z}^{(l)} = h(\boldsymbol{a}^{(l)}) = \begin{pmatrix} h(a_1^{(l)}) \\ h(a_2^{(l)}) \\ h(a_3^{(l)}) \end{pmatrix}$$

ニューラルネットワークを数式で

- 2層ニューラルネットワークを数式で書くと
  1. アフィン変換
  2. 非線形変換
  3. アフィン変換
  4. 非線形変換



$$\boldsymbol{y} = f(\boldsymbol{x}) = h\left(W^{(2)}h\left(W^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}\right)$$

どんな問題を解くか

- ざっくり「回帰問題」と「分類問題」
  - 回帰問題
    - 入力から数値の予測を行う
  - 分類問題
    - 入力のデータから分類を行う
- 解く問題に合わせて出力層の活性化関数を変更
  - 恒等関数
  - ソフトマックス関数

$$y_k = \frac{\exp(a_k)}{\sum_{i=0}^{n} \exp(a_i)}$$

ソフトマックス関数の特徴

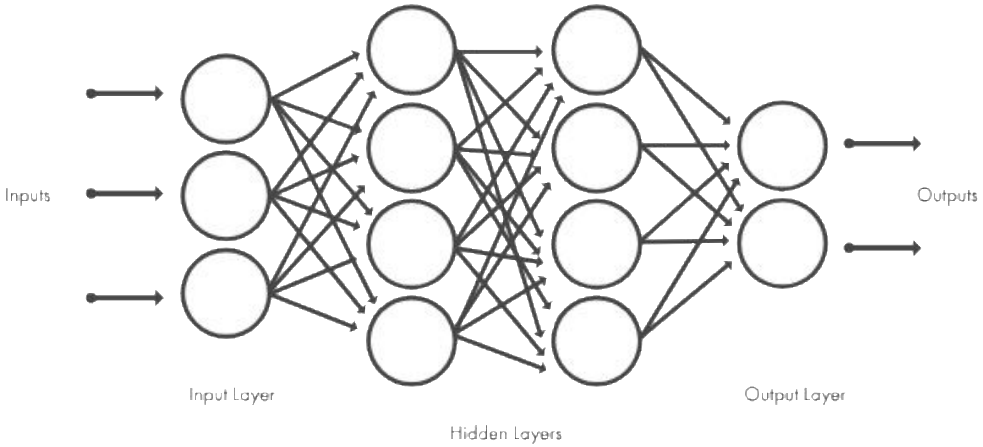
- ソフトマックス関数の出力は(0,1)
- 出力の総和は1
- 出力を確率とみなせる

$$y_k = \frac{\exp(a_k)}{\sum_{i=0}^{n} \exp(a_i)}$$

例）入力データが「数字の5」である確率は0.8

順伝播

- 入力層にある値を入力し、ニューロンの信号を逐次的に計算し階層的に出力層へ伝えること



- 要は関数にデータを入れて出力を得ること

$$\boldsymbol{y} = f(\boldsymbol{x}) = h\left(W^{(2)}h\left(W^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}\right)$$

まとめ

- ニューラルネットワークでは活性化関数として非線形関数を使う
- 非線形関数にはシグモイド関数やReLU関数など微分可能な関数を使う
- 機械学習の問題はざっくりと「回帰問題」と「分類問題」に分けられる
- 出力層には、回帰問題に恒等関数、分類問題にソフトマックス関数を使う
- ニューラルネットワークで入力から出力を得る計算手続きを「順伝播」と呼ぶ

#4 ニューラルネットワークの学習

学習

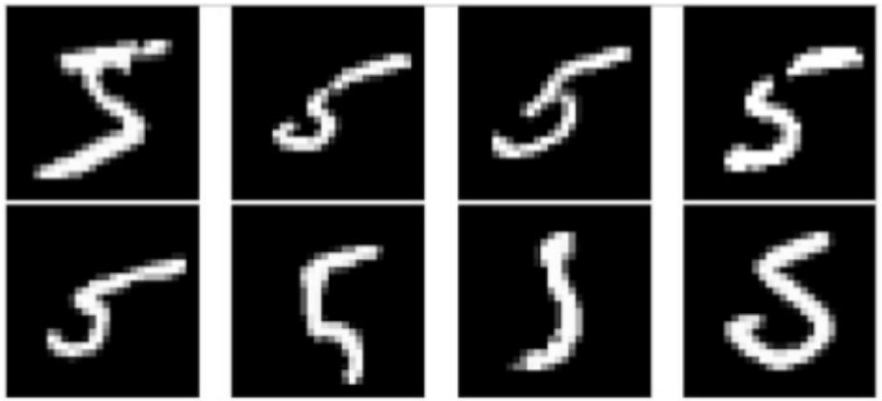
- 学習データから問題を解くのに最適なパラメータの値を自動で獲得すること
  - Wとかbとかを決める

$$\boldsymbol{y} = f(\boldsymbol{x}) = h\left(W^{(2)}h\left(W^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}\right)$$

- 問題を解けていない度を測る損失関数を導入
- 損失関数の値が小さくなる（問題がまぁ解ける）パラメータを探し出す

手書き数字の分類の例

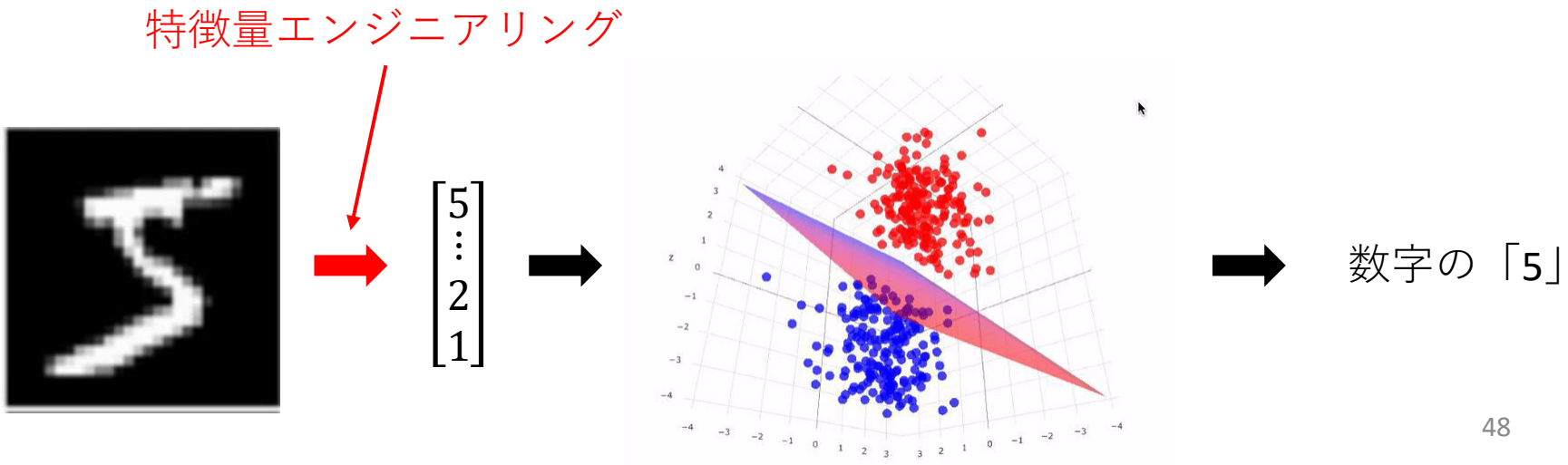
- 数字の「5」を画像から認識する問題を考える



- 数字の「5」っていろんな形がある
- 「5」であることのルールを決めるのは大変
- ヒューリスティックは無謀

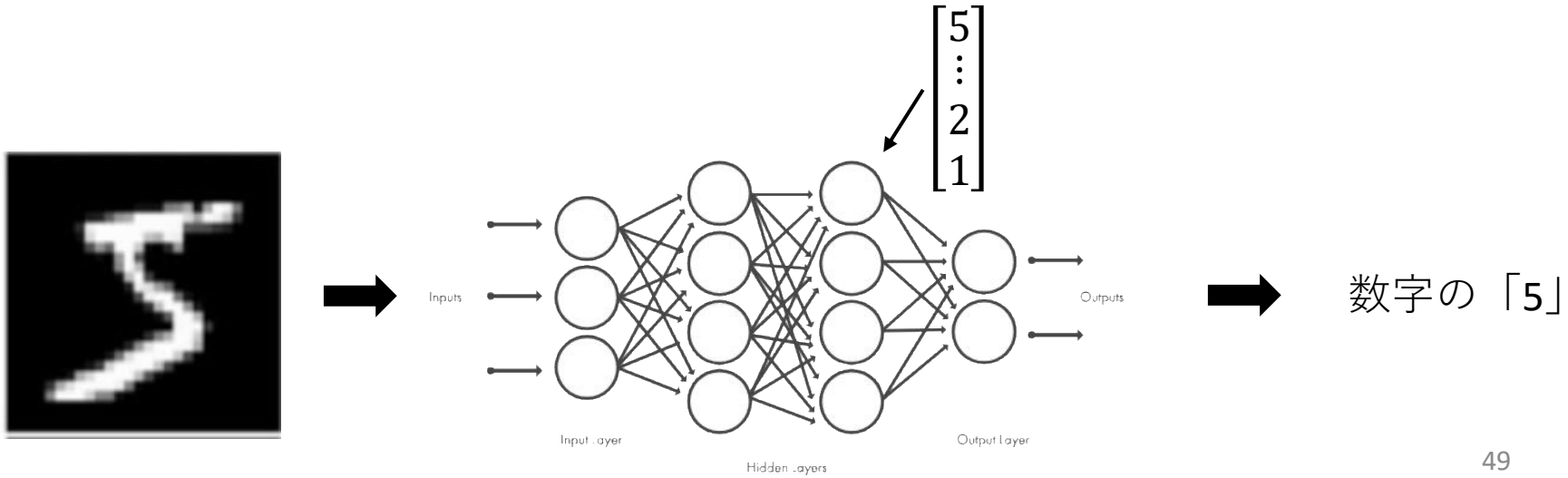
特徴量＆機械学習のアプローチ

- 特徴量とは入力データの本質を良く表現する何かしらの値（スカラー、ベクトル）
- 特徴量を識別器（SVMやKNN）に与え学習
- 特徴量の作り方は問題によって大きく変わる
- 複雑なデータ構造では人手で特徴量が作れない



ニューラルネットワークのアプローチ

- 画像から答えを得るための特徴量を学習する
  - 表現学習とも呼ばれる
- 表現を用いてタスクを解く方法も学習
- 生データから目的の出力を得られる
  - end-to-end



汎化と過学習

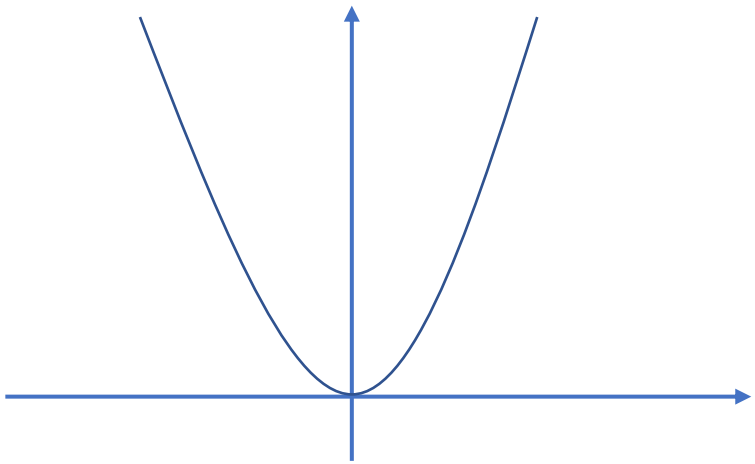
- 汎化性能
  - 見たことのない（学習したことのない）データに対するパフォーマンス
- 過学習
  - 手持ちのデータ（学習データ）だけ過度にパフォーマンスが高い状態
  - 汎化性能が低い状態
- 学習の真の目的は、過学習を避け、汎化性能を向上させること

汎化性能を図るには？

- 手持ちのデータを訓練データとテストデータに分ける
- 訓練データとテストデータの重複を避ける
  - テストデータ＝見たことないデータ
- 訓練データだけを使って学習
- テストデータでモデルを評価

損失関数

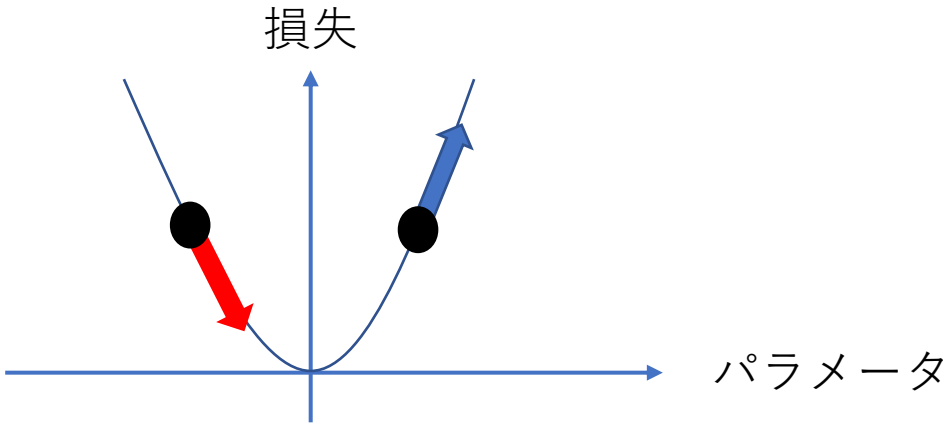
- 問題の解けていない度を定量化する関数
- ニューラルネットワークの学習とは、損失関数を最小化するパラメータを決定すること
- 平均二乗誤差
  - 正解から離れれば離れるほど値が大きくなる



$$MSE = \frac{1}{2}\sum_{k=1}^{N}(y_k - \hat{y}_k)$$

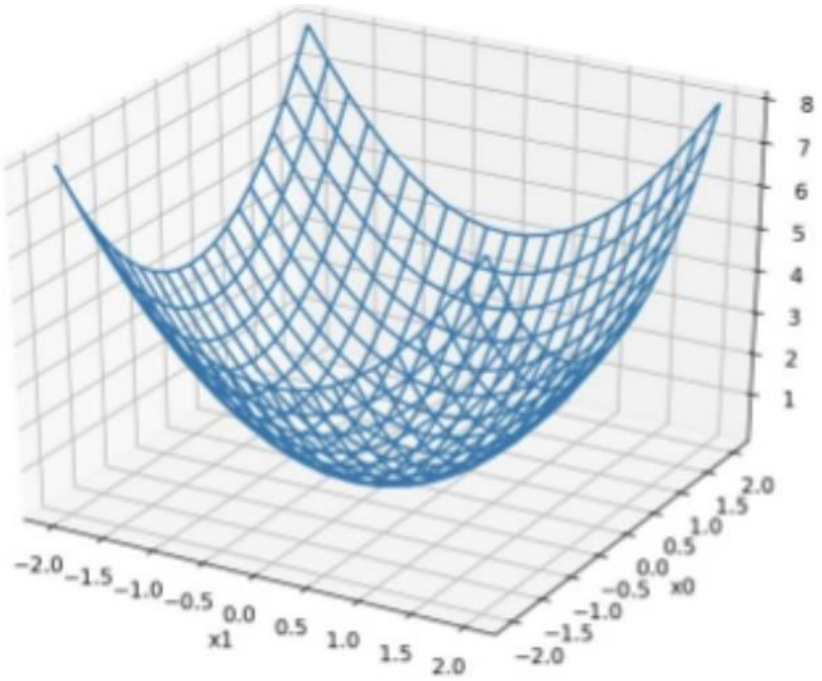
損失関数と学習の関わり

- 損失関数を最小化するため、パラメータの微分を計算し、その微分の値を手掛かりにパラメータの値を徐々に調整する
- パラメータの損失関数に対する微分の意味
  - 「パラメータの値を少し変化させると、損失関数の値がどのように変化するか」
  - 微分が負なら正の方向に更新（逆も然り）



勾配

- 微分の多次元版（偏微分を要素とするベクトル）



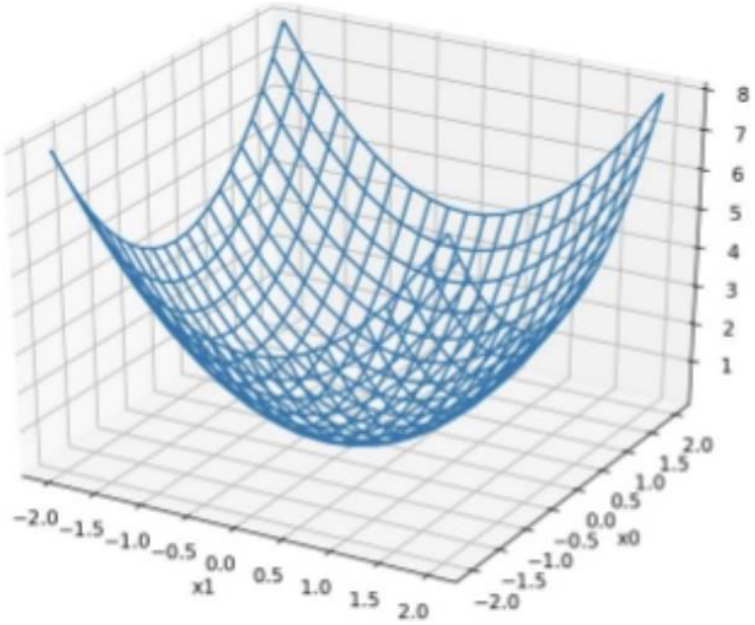
$$f(x_0, x_1) = x_0^2 + x_1^2$$

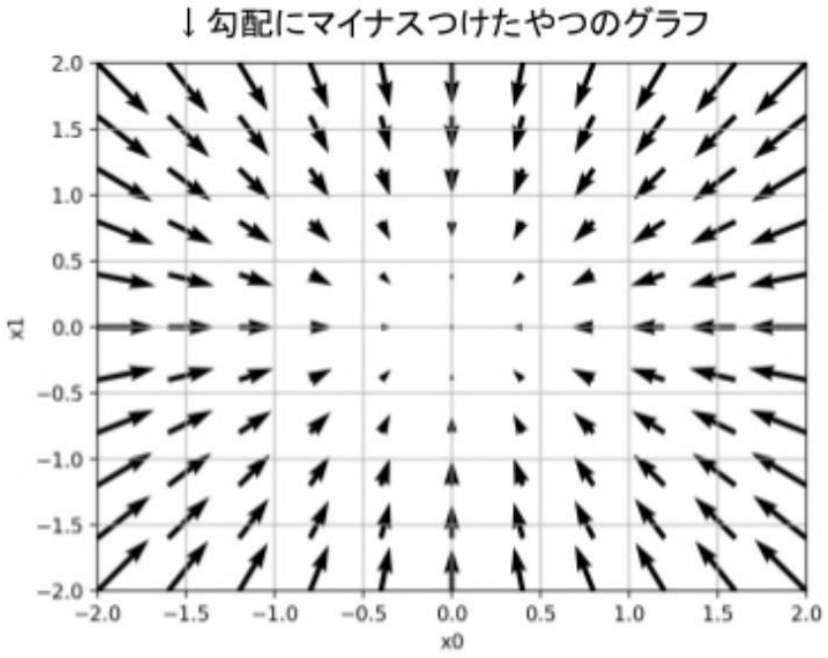
$$\begin{cases} \dfrac{\partial f}{\partial x_0} = 2x_0 \\ \dfrac{\partial f}{\partial x_1} = 2x_1 \end{cases}$$

$$\nabla f = \left(\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1} \right)$$

勾配の可視化

- 負方向の勾配は損失関数が小さくなる方向





勾配（降下）法

- 勾配をうまく利用して損失関数の値が最小（できるだけ小さな値）となるパラメータを探す
- 注意！
  - 各地点において（局所的にみて）関数の値を最も減らすのが負方向の勾配
  - 大域的最小値である保証はない！
- パラメータを負方向の勾配に逐次的更新する

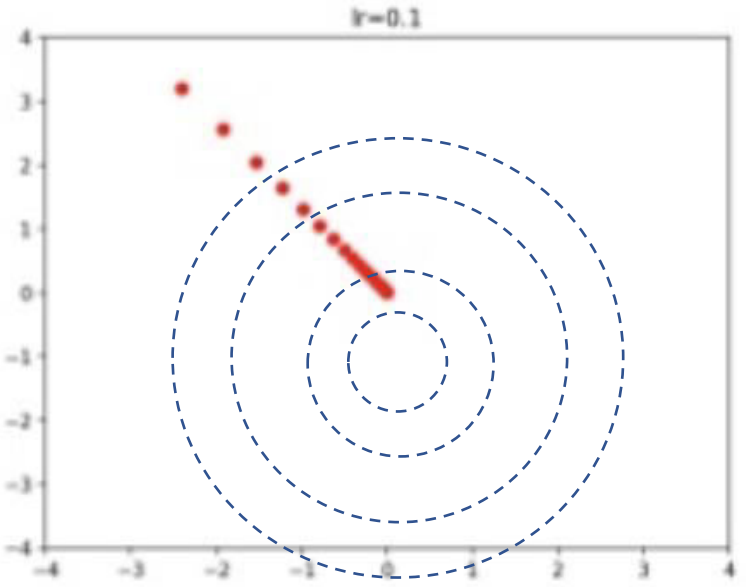
勾配（降下）法を数式で

$$f(x_0, x_1) = x_0^2 + x_1^2$$

$$\nabla f = \left(\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1} \right)$$

学習率

$$\begin{cases} x_0 = x_0 - \eta \dfrac{\partial f}{\partial x_0} \\ x_1 = x_1 - \eta \dfrac{\partial f}{\partial x_1} \end{cases}$$

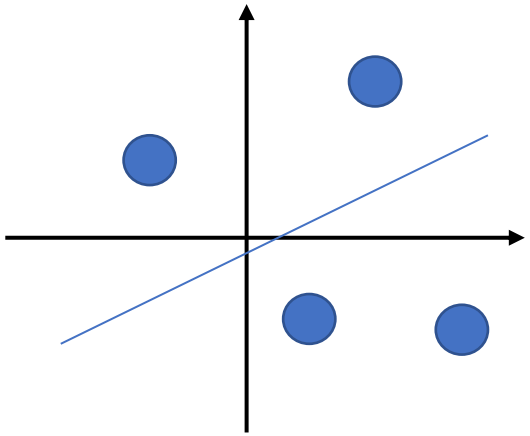


ニューラルネットワークに対する勾配

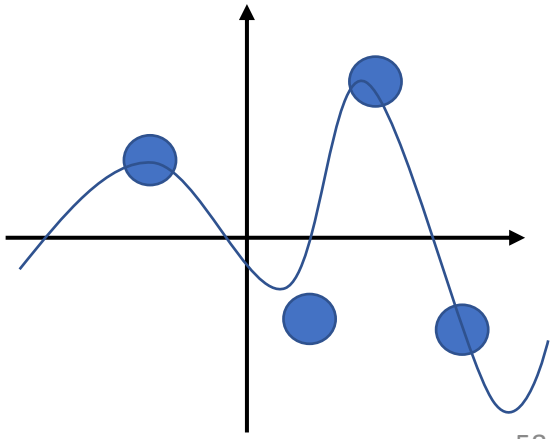
- パラメータθ（重みとバイアス）に対する損失関数Lの勾配

$$\boldsymbol{y} = f(\boldsymbol{x}) = h\left(W^{(2)}h\left(W^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}\right)$$

$$\frac{\partial L}{\partial \theta} = \left\{\frac{\partial L}{\partial W^{(2)}}, \frac{\partial L}{\partial W^{(1)}}, \frac{\partial L}{\partial \boldsymbol{b}^{(2)}}, \frac{\partial L}{\partial \boldsymbol{b}^{(1)}}\right\}$$



勾配法で
パラメータを調整



どうやって勾配を計算すんの

- 数値微分
  - 単純な方法
  - 実際はやりません

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- 解析的に微分を計算する
  - 誤差逆伝播法（次章）
  - ニューラルネットワークが表現する関数と損失関数が微分可能である必要
    （YESシグモイドNOステップ）

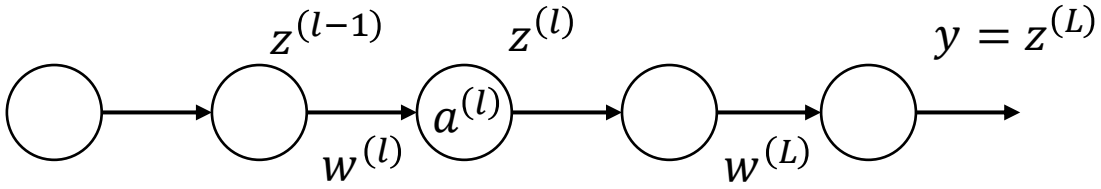
まとめ

- ニューラルネットワークは、特徴量の学習（表現学習）と目的のタスクを解く合わせ技
  - end-to-end
- ニューラルネットワークの学習は、損失関数を指標として、損失関数の値が小さくなるように、重みパラメータを更新する
- 重みパラメータを更新する際には、重みパラメータの勾配を利用して、勾配方向に重みの値を更新する作業を繰り返す

#5 誤差逆伝播法

勾配法を考えていく

- 各層が1つのユニットだけからなるニューラルネットを考える
- w(l)の変動は続く層の出力を変動させる
- w(l)を勾配法によって更新することを考えると、巨大な合成関数の微分を計算する必要がある



$$y = h\left(w^{(L)}h\left(w^{(L-1)}h(\cdots w^{(l+1)}h(w^{(l)}h(\cdots h(x))))\right)\right)$$

w(l)について考える

- w(l)を勾配法で更新するとは

$$w_{new}^{(l)} = w_{old}^{(l)} - \eta \frac{\partial L}{\partial w_{old}^{(l)}}$$

- w(l)の変動はa(l)の値に直接影響を与える
- δ(l)はδ(l+1)と関連

$$a^{(l)} = w^{(l)} z^{(l-1)} \qquad z^{(l)} = h(a^{(l)})$$

$$\frac{\partial L}{\partial w^{(l)}} = \frac{\partial L}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial w^{(l)}} = \frac{\partial L}{\partial a^{(l)}} z^{(l-1)} \equiv \delta^{(l)} z^{(l-1)}$$

$$\delta^{(l)} = \frac{\partial L}{\partial a^{(l+1)}} \frac{\partial a^{(l+1)}}{\partial a^{(l)}} = \delta^{(l+1)} w^{(l+1)} h'(a^{(l)})$$

順伝播と逆伝播

- 順伝播
  - l-1層からl層に向けてアフィン変換と非線形変換を繰り返す
- 逆伝播
  - l+1層からl層に向けて損失関数の勾配δを漸化的に計算する

$$\delta^{(l)} = \frac{\partial L}{\partial a^{(l+1)}} \frac{\partial a^{(l+1)}}{\partial a^{(l)}} = \delta^{(l+1)} w^{(l+1)} h'(a^{(l)})$$

$$\boldsymbol{x} \to z^{(1)} \to z^{(2)} \to \cdots \to z^{(L-1)} \to z^{(L)} = y$$

$$\delta^{(1)} \leftarrow \delta^{(2)} \leftarrow \cdots \leftarrow \delta^{(L-2)} \leftarrow \delta^{(L-1)} \leftarrow \delta^{(L)}$$

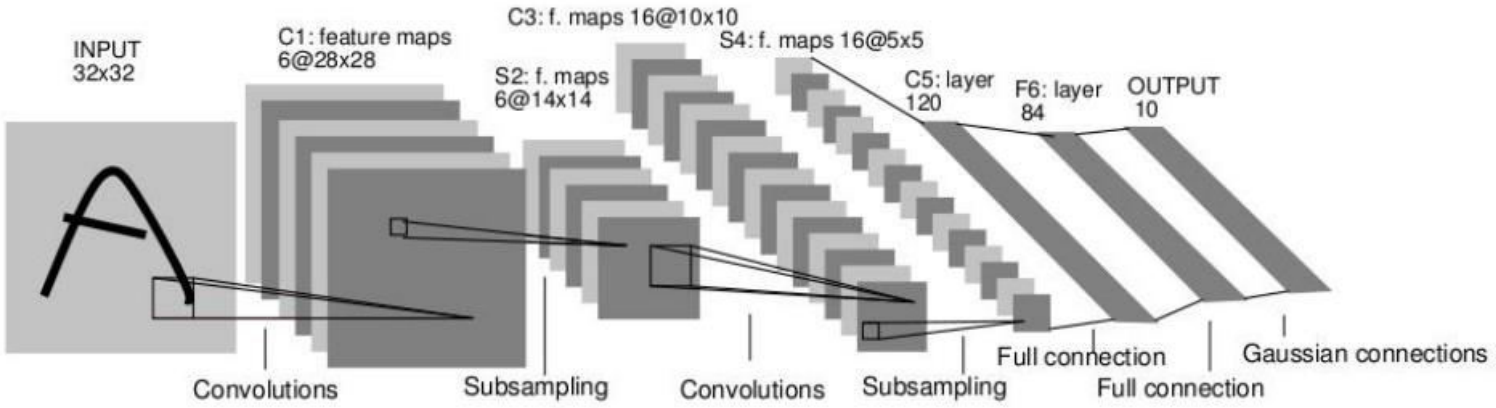
まとめ

- 誤差逆伝播法は、最終層から誤差の勾配を漸化的に計算することでパラメータの勾配を計算する手法

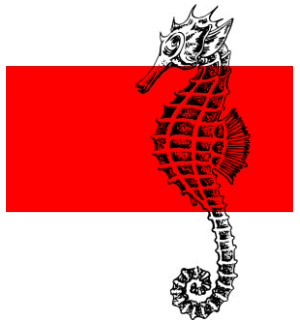
#7 畳み込みニューラルネット

畳み込みニューラルネット

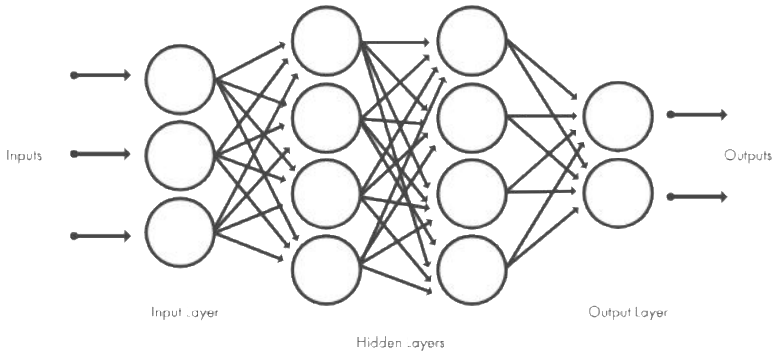
- Convolutional Neural Network (CNN)
- 画像分野で広く使用
  - 音声・時系列などよく使われている
- 畳み込み層とプーリング層



全結合

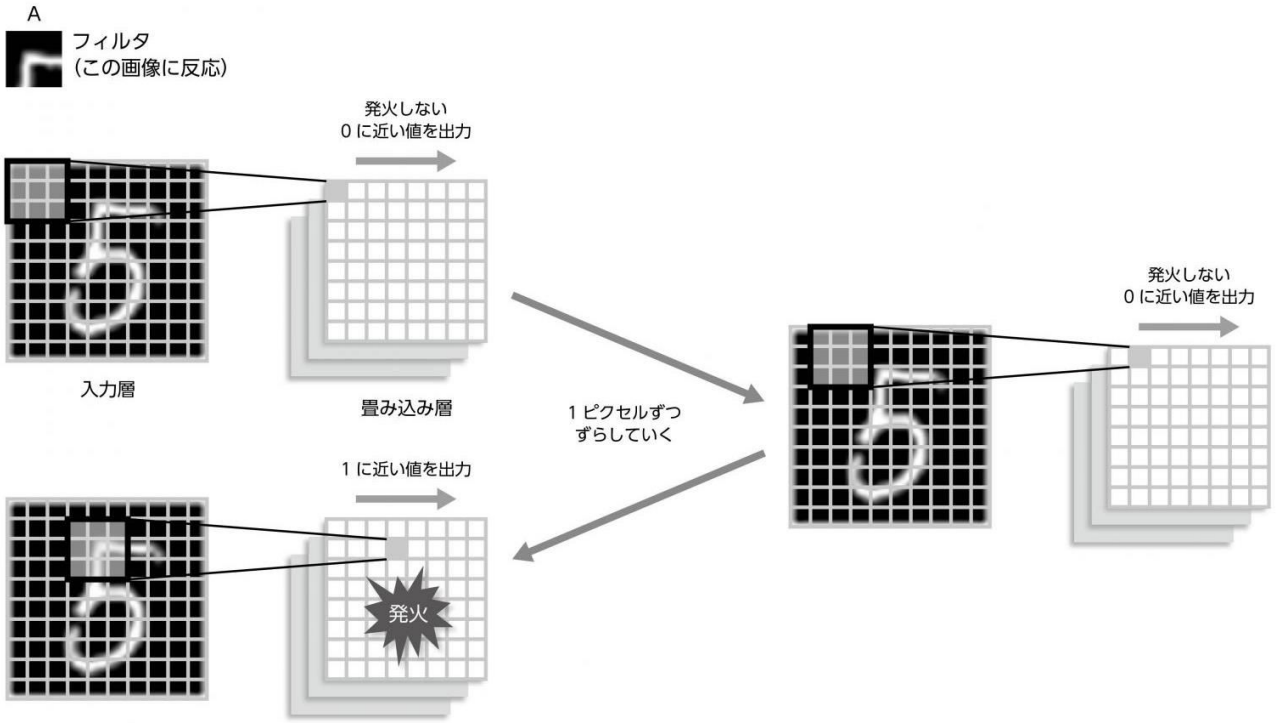


- さっきまで説明していたのは**全結合**
  - 入力データ全体を見て特徴量を作る
  - データの形状が無視される
- 画像は大切な空間的情報が含まれている
- 畳み込みは形状を維持する
  - 空間的情報に意味があると仮定
  - 入出力データを**特徴マップ**とも呼ぶ



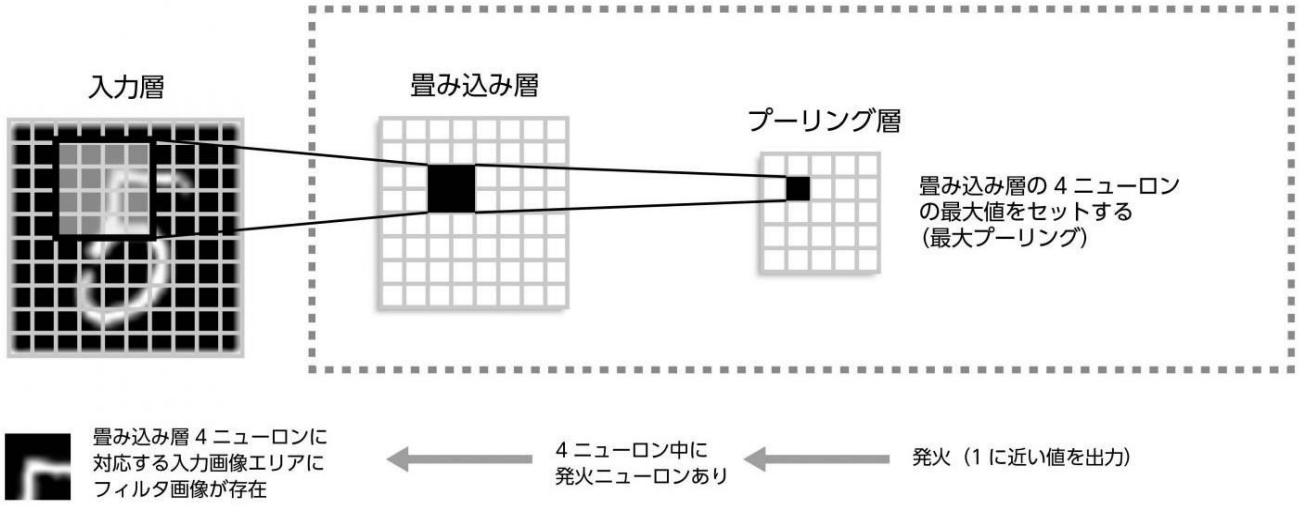
畳み込み層

- 重み＝フィルタ（カーネル）のパラメータ



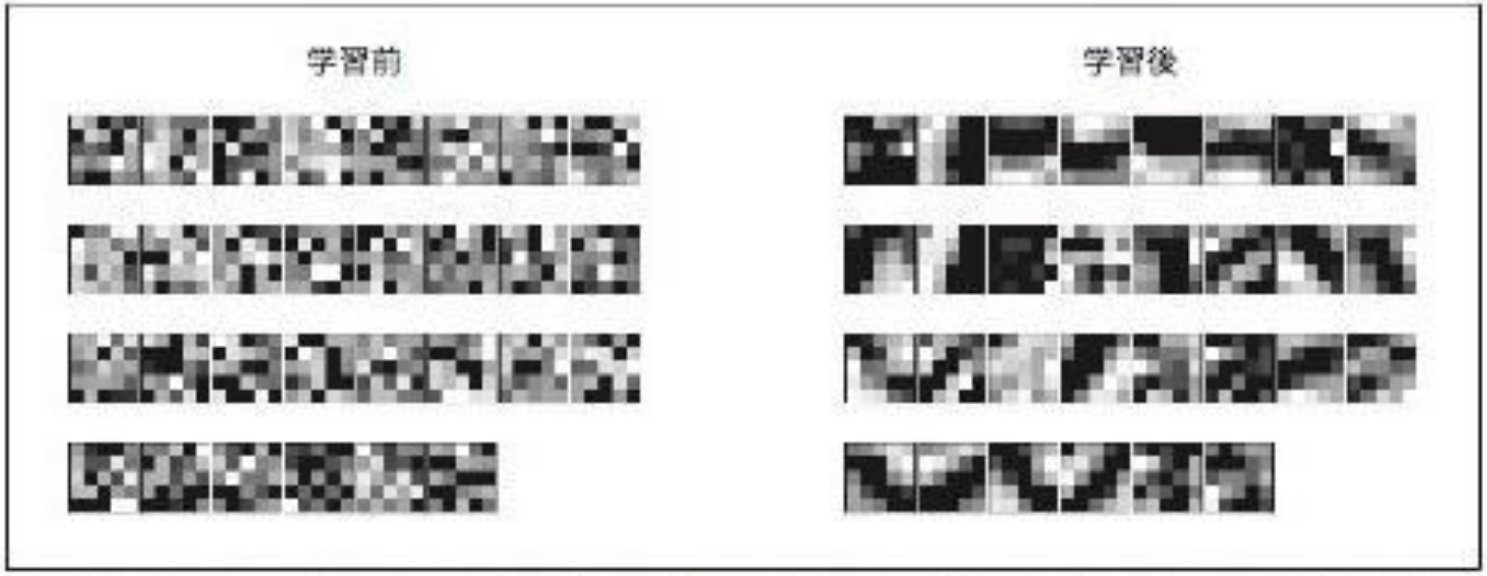
プーリング層

- 画像サイズを小さくする
- 入力のズレを吸収する
  - 猫の画像がちょっとズレても猫



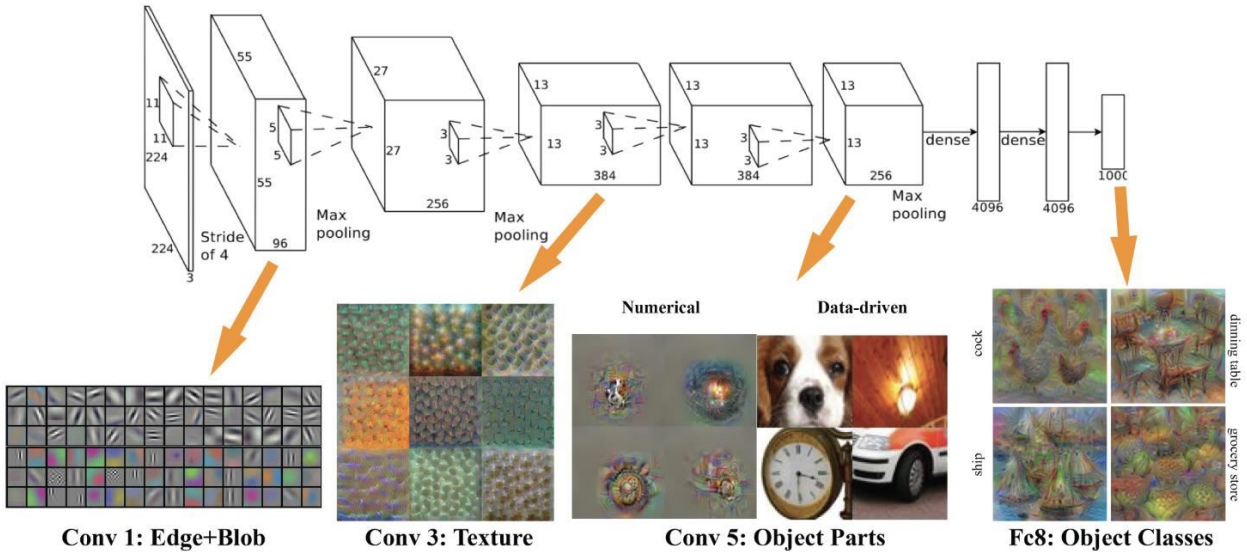
畳み込みの可視化

- フィルタ（カーネル）は学習を経て規則性のあるフィルタへと更新されている



階層構造による情報抽出

- 層が深くなるに従って抽出される情報はより抽象化される
  - エッジ、ブロブ
  - 犬の顔



まとめ

- 畳み込み層
  - フィルタ
  - 局所的な情報に意味があると仮定
- プーリング層
  - 特徴マップのサイズの縮小
  - 入力のズレの吸収
- 層が深くなると高度な情報が抽出される

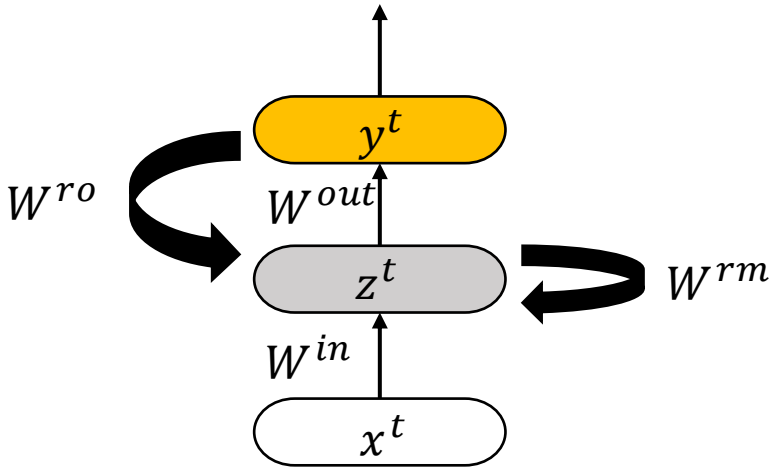
再帰型
ニューラルネット

時系列データ

- 深層学習をうまく働かせるカギは「いかにデータに適したネットワーク構造をデザインできるか」
  - CNNは画像特化
- 動画や文章、会話のような時系列データをニューラルネットで扱うには？
  - 前の内容に依存
  - 系列長がまちまち

再帰型ニューラルネット

- 時系列データの情報をうまく扱うには、過去の情報を集約し保持する必要がある
- Recurrent Neural Network (RNN)



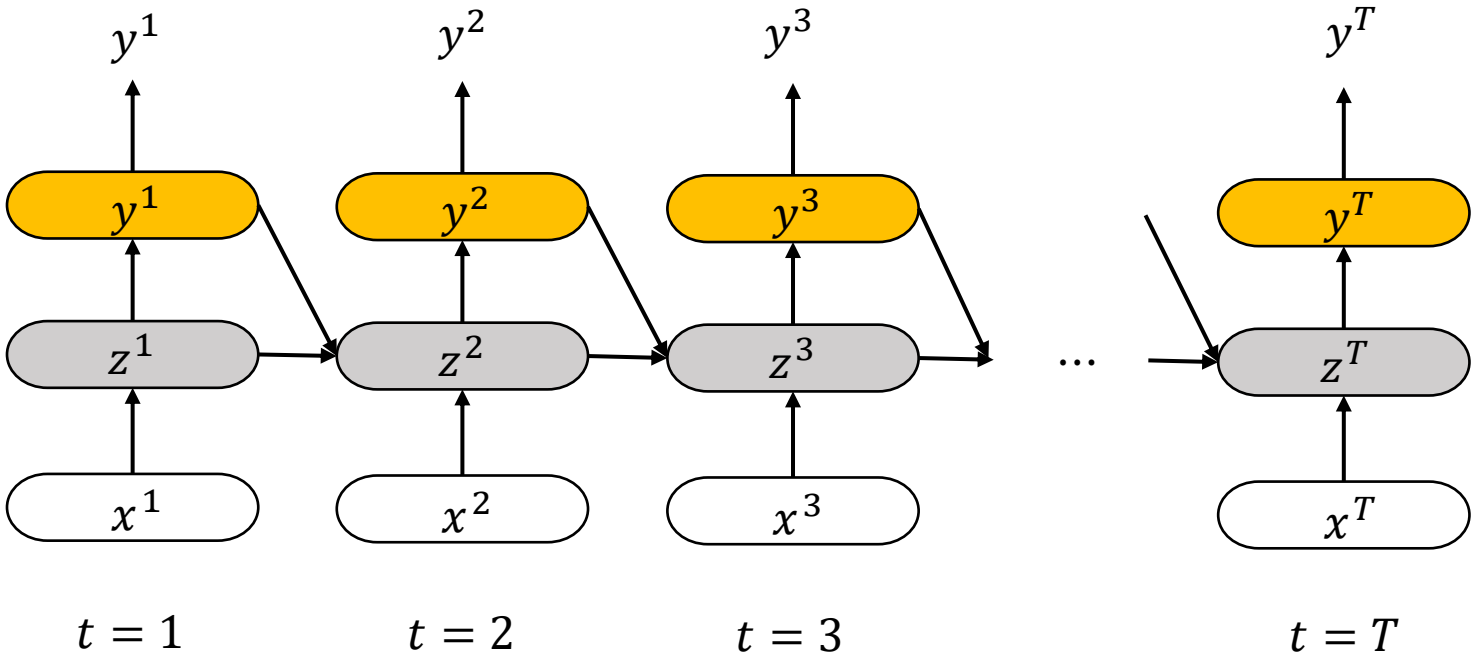
$$a^t = W^{in}x^t + W^{rm}z^{t-1} + W^{ro}y^{t-1}$$

$$z^t = h(a^t)$$

$$y^t = h(W^{out}z^t)$$

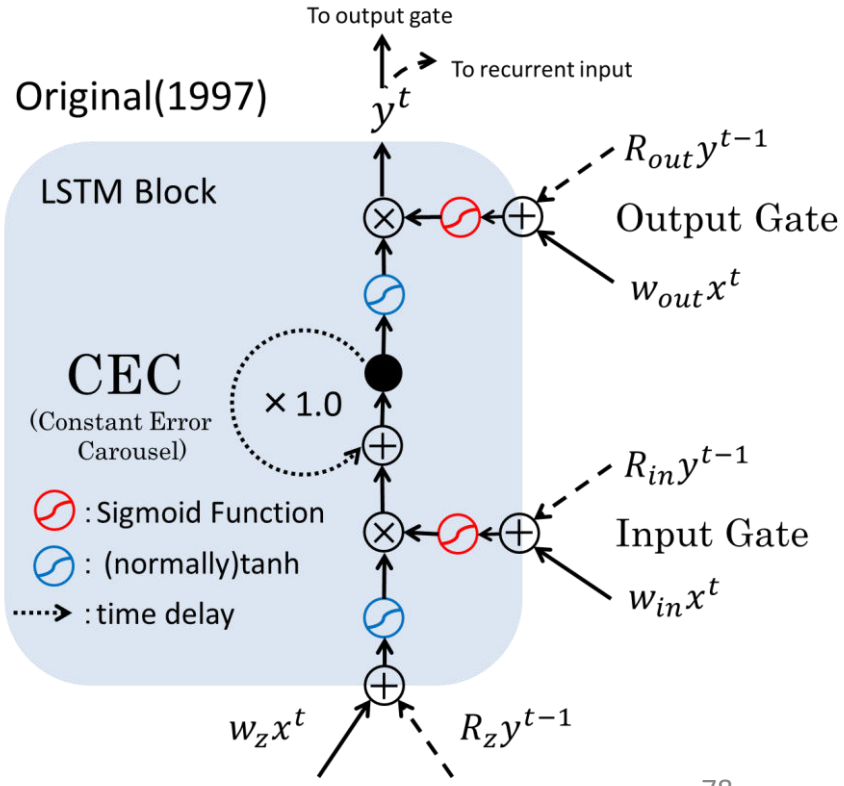
ネットワークの展開

- RNNを展開して処理を理解する
- 入力は逐次的に入力・処理される



長・短期記憶

- LSTM (Long Short Term Memory)
- 現在でもよく使われるRNNの一種
- 詳細は省略



さいごに

普遍性定理

- ネットワークが十分な数の隠れユニットを持つ場合、線形の出力層と非線形の活性化関数を持つ隠れ層が少なくとも1つ含まれる順伝播型ネットワークは、どんなボレル可測関数でも任意の精度で近似できる。
- ひらたくいえば、順伝播型ニューラルネットワークは任意の関数を任意の精度で近似できる！
- 隠れ層が1つでもこの定理は成り立つ（深層でなくて良い）

普遍性定理と深層学習

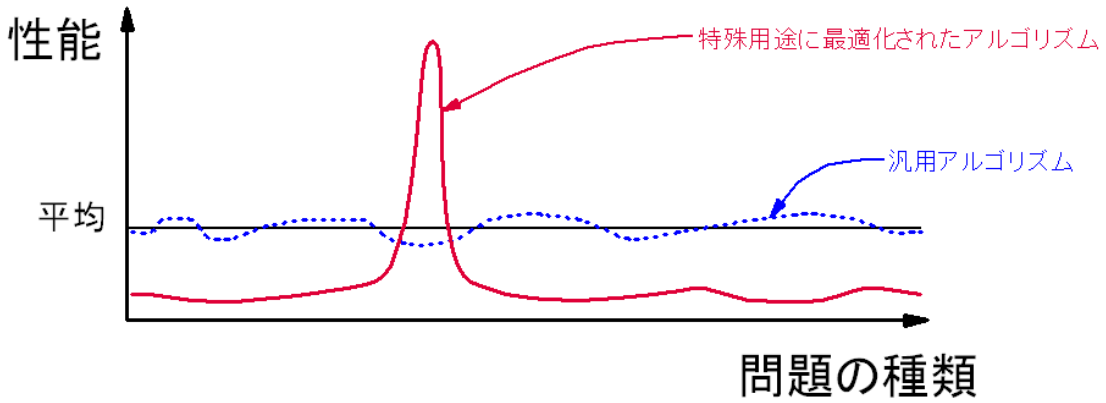
- 普遍性定理は、どんな関数を学習するかに関わらず、大きなMLPであればその関数を表現できるということを示している。
- 訓練アルゴリズムがその関数を学習できるかは保証されていない！
  - パラメータ空間が広大すぎて探索不可
  - 過学習で間違った関数を選択する可能性
- 多くの場合、深いモデルを利用すると、目的の関数を表現するために必要なパラメータの数を減らし、汎化誤差を減らすことができる。

層を深くすることのモチベーション

- 統計的に
  - 学習したい関数が複数の単純な関数で構成される
- 表現学習的に
  - もっとも潜在的で意味のある特徴量は単純な特徴量を組み合わせることで得られる
- コンピュータサイエンス的に
  - 学習したい関数が複数のステップからなるコンピュータプログラムであり、その各ステップは直前のステップの出力を利用する

ノーフリーランチ定理（タダ飯はねぇ）

- あらゆる問題で性能の良い汎用最適化戦略は理論上不可能
- ある戦略が他の戦略より性能が良いのは、現に解こうとしている特定の問題に対して特殊化（専門化）されている場合のみである。

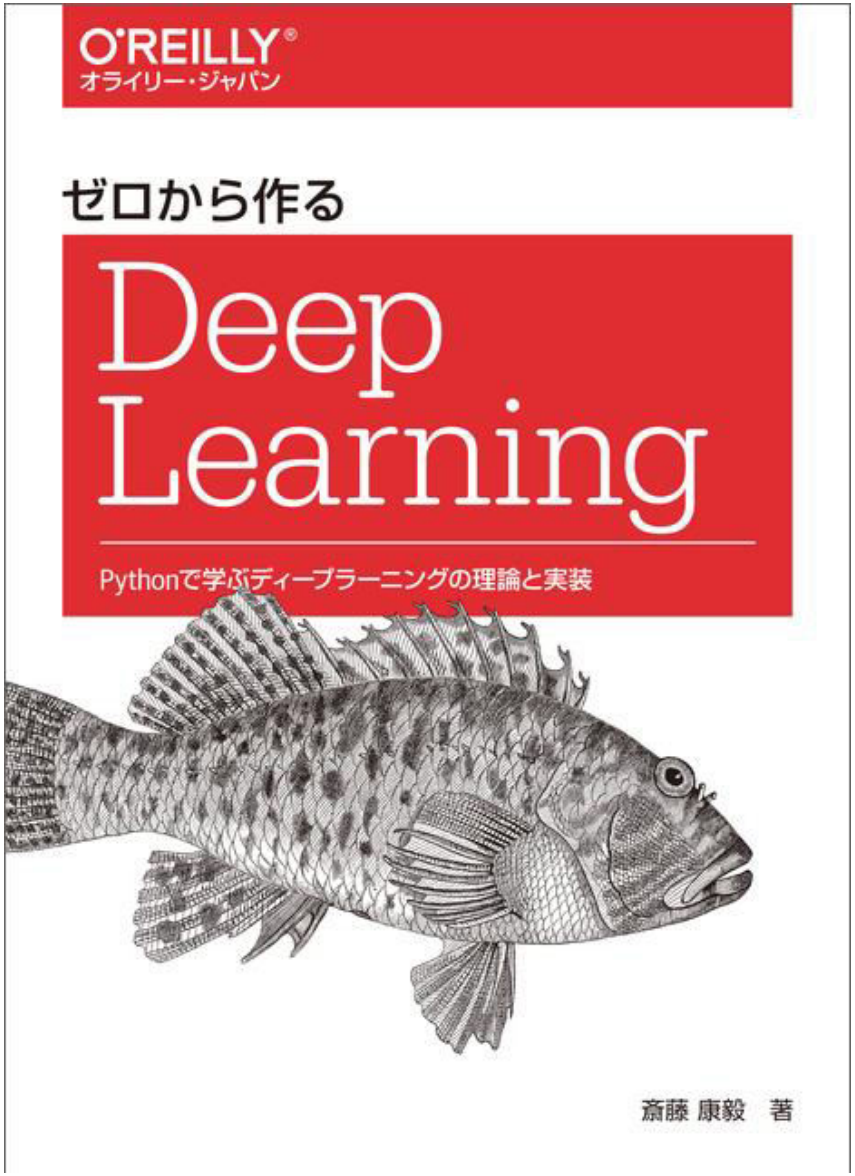


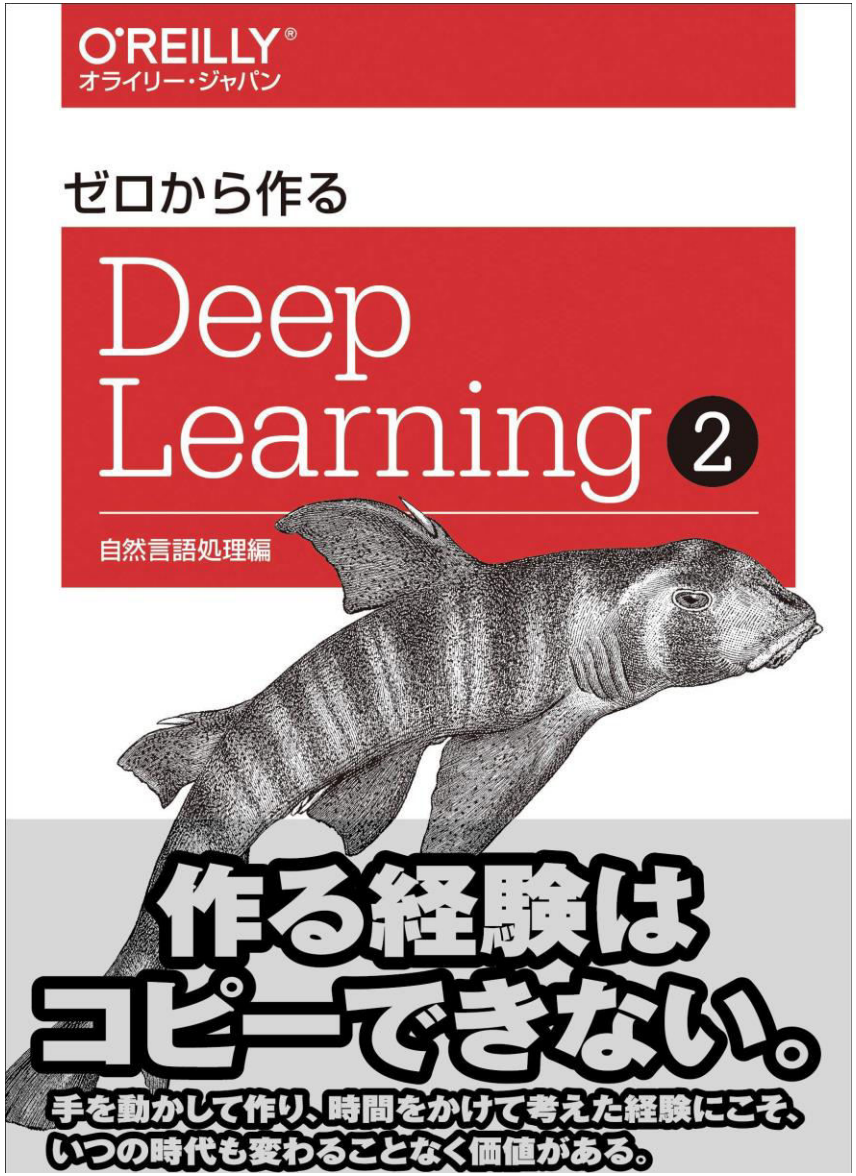
普遍性定理＆ノーフリーランチ定理

- 普遍性定理
  - 順伝播ニューラルネットワークは任意の関数を任意の精度で近似できる
- ノーフリーランチ定理
  - 普遍的に優れた機械学習アルゴリズムは存在しない
- ニューラルネットワークは関数を表現するための普遍的なシステム
  - ある関数が与えられた時に、その関数を任意の精度で近似できる順伝播ネットワークが存在する

おすすめ書籍

わかりやすい

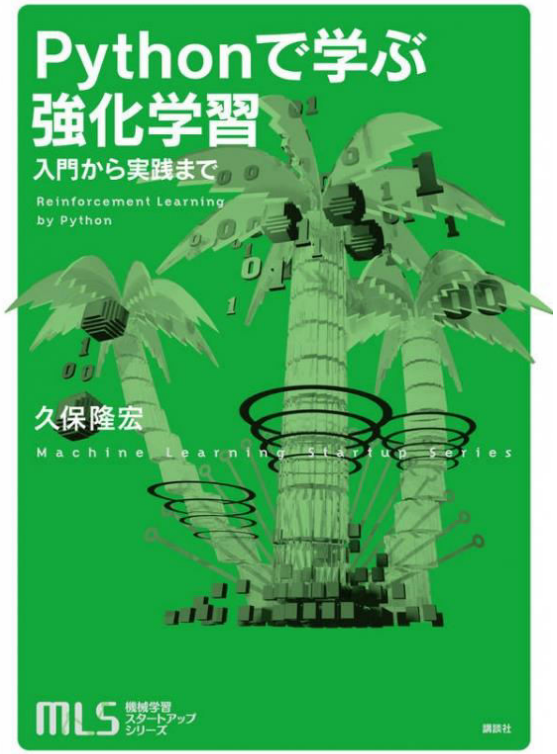

O'REILLY
オライリー・ジャパン
ゼロから作る
Deep Learning
Pythonで学ぶディープラーニングの理論と実装
斎藤 康毅 著


O'REILLY
オライリー・ジャパン
ゼロから作る
Deep Learning 2
自然言語処理編
作る経験はコピーできない。
手を動かして作り、時間をかけて考えた経験にこそ、
いつの時代も変わることなく価値がある。

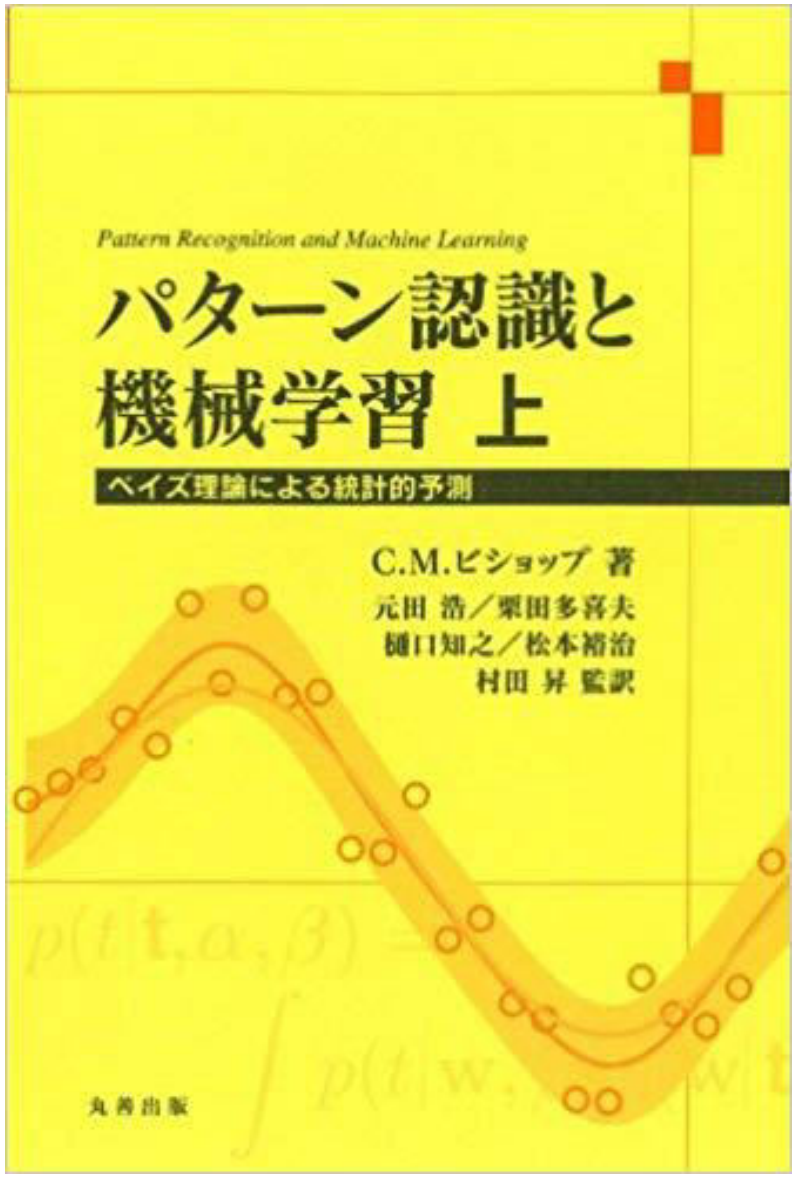

Python Machine Learning 2nd Edition
impress
top gear
［第2版］
達人データサイエンティストによる理論と実践
Python
機械学習プログラミング
Sebastian Raschka / Vahid Mirjalili 著
株式会社クイープ 訳
福島 真太朗 監訳
ML本ベストセラーの第2版!
機械学習の全般を力バー
◎各手法の考え方と数学的背景を解説
◎scikit-learnやTensorFlowで実践コーディング
◎CNNやRNNについても新たに解説
インプレス


詳解 ディープラーニング
TensorFlow・Kerasによる時系列データ処理
巣籠 悠輔 [著]
ゼロから丁寧に説明。
[基本から応用・理論と実装]
ディープラーニング、ニューラルネットワークについて ディープラーニング向けの
Pythonライブラリ"TensorFlow" および"Keras"を用い丁寧に解説。
[予備知識なしでOK!]
時系列データ処理のためのディープラーニングのアルゴリズムに焦点を当てます。
マイナビ


ベイズ推論による
機械学習
入門
Introduction to Machine Learning
by Bayesian Inference
須山敦志 著
杉山 将 監修
Machine Learning Startup Series
どこまでも分かりやすい!
自分が学生の頃にこのような教科書が
あったら良かったのに、と感じさせる一冊
理化学研究所革新知能統合研究センター センター長
東京大学大学院新領域創成科学研究科 教授
杉山 将
MLS 機械学習 スタートアップ シリーズ
講談社


Pythonで学ぶ
強化学習
入門から実践まで
Reinforcement Learning
by Python
久保隆宏
Machine Learning Startup Series
MLS 機械学習 スタートアップ シリーズ
講談社


これならわかる
深層学習
入門
Introduction to Deep Learning
瀧 雅人
Machine Learning Startup Series
MLS 機械学習 スタートアップ シリーズ
講談社


Pattern Recognition and Machine Learning
パターン認識と
機械学習 上
ベイズ理論による統計的予測
C.M.ビショップ 著
元田 浩／栗田多喜夫
樋口知之／松本裕治
村田 昇 監訳
丸善出版


Pattern Recognition and Machine Learning
パターン認識と
機械学習 下
ベイズ理論による統計的予測
C.M.ビショップ 著
元田 浩／栗田多喜夫
樋口知之／松本裕治
村田 昇 監訳
待望の邦訳
PATTERN RECOGNITION AND MACHINE LEARNING
CHRISTOPHER M. BISHOP
英語版も
息ちら刷
サポートベクトルマシン，ベイジアンネット，変分ベイズ，
MCMC，隠れマルコフモデル，ブースティング，...
ベイズ理論に基づいた
統一的な視点から
機械学習とパターン認識の
様々な理論や手法を解説
シュプリンガー・ジャパン