

現場で使えるディープラーニング基礎講座

DAY1

SkillUP AI

本講座の事前準備

- 必要環境

- Anaconda3系の指定のバージョン(別途連絡)。
- 開発環境にはJupyter Notebook、言語はPythonを使います。
- DAY1教材に含まれる0_preparation.ipynbの手順に沿って、環境を構築してください。
- GPUは必須ではありません。

- 参考図書

- ① 『深層学習』(松尾豊監修、KADOKAWA)
- ② 『ゼロから作るDeep Learning』(斎藤康毅、オライリーJ)
- ③ 『深層学習』(岡谷、講談社)
- ④ 『画像認識』(原田、講談社)
- ⑤ 『実装ディープラーニング』(藤田など、オーム社)
- ⑥ 『イラストで学ぶディープラーニング』(山下、講談社)
- ⑦ 『TensorFlow活用ガイド』(下田など、技術評論社)
- ⑧ 『詳解ディープラーニング』(巢籠、マイナビ出版)
- ⑨ 『ゼロから作るDeep Learning② -自然言語処理編-』(斎藤康毅、オライリーJ)

本講座での用語

- 本講座では、ディープラーニングまわりの用語を以下のように扱います。
 - DL = Deep Learning = ディープラーニング = 深層学習
 - 機械学習の1分野を示す言葉として用いる。
 - NN = Neural Network
 - DNN = Deep Neural Network
 - 深層学習のモデル全般を示す言葉として用いる。特に全結合型のNNを指す。
 - CNN = Convolutional Neural Network
 - RNN = Recurrent Neural Network

本講座での表記

- 本講座では、数式などの表記を以下のように扱います。
 - a : スカラー
 - \mathbf{a} : ベクトル
 - A : 行列
 - \mathbf{A} : テンソル
 - I : 単位行列
 - $\{0,1\}$: 0と1からなる集合
 - $A_{i,j}$: 行列 A の i 行 j 列の要素
 - A^T : 行列 A の転置行列
 - $A \odot B$: 行列 A と行列 B の要素ごとの積(アダマール積)
 - $\frac{\partial y}{\partial x}$: y の x に関する偏微分
 - $\log x$: x の自然対数
 - $\sigma(x)$: シグモイド関数
 - $\|\mathbf{x}\|_p$: ベクトル \mathbf{x} の L_p ノルム
 - $\|\mathbf{x}\|$: ベクトル \mathbf{x} の L_2 ノルム

本講座の方針 1/2

- 本講座は、動画講義、Notebook演習、対面講義、通し課題、知識テストで構成されます。
- 対面講義では、動画講義に関する深掘りとグループワークを行います。
 - 対面講義は、動画講義を視聴している前提で進むので、必ず動画講義を視聴してから対面講義にお越し下さい。
- 通し課題では、DAY1-DAY8までを通して、1つの課題に取り組んで頂きます。
- 知識テストは、数学知識テスト、ML知識テスト、DL知識テストの3種類あり、いつでも何度でも受けることができます。
 - 講座の後半は、予習と通し課題に時間が取られますので、知識テストはなるべく早めに取り組んでおきましょう。

本講座の方針 2/2

- ディープラーニングにおける各種手法の理論や性質を正しく理解することを目指します。
- 各種手法の理論や性質を正しく理解するには、手を動かすことが最も重要であると考え、ニューラルネットワークをNumpyベースで実装していきます。
 - TensorFlowのようなフレームワークは、ほとんど扱いません。
- 講座を通した課題に取り組むことによって、現場での実践力を身につけていきます。

Notebook演習について

- `****_trainee.ipynb`という名称のNotebookファイルは穴埋め形式になっています。
- その解答例ファイルは、`_trainee`が末尾に付いていない同名のファイルです。
- E資格試験では、Pythonで記述されたコードの穴埋め問題が出題されるため、E資格試験を受験される方は、穴埋め問題にもしっかり取り組んでください。
 - 穴埋め部分に何を入れたらいいかが分からないところは、一度解答例ファイルを覗いてから、もう一度`****_trainee.ipynb`に挑戦してみましょう。
 - この穴埋め問題は、E資格試験の出題形式を模しているので、必ず本番までに解けるようになっておきましょう。

本講座の構成

DAY1

- ディープラーニング講座を通しての課題
- ディープラーニング基礎 前半
 - パーセプトロン
 - ニューラルネットワーク
 - 活性化関数
 - 順伝播計算
 - 出力層の設計
 - 予測関数
 - バッチ処理
 - 損失関数

DAY2

- ディープラーニング基礎 後半
 - ミニバッチ学習
 - 微分
 - 最急降下法
 - 勾配法
 - 誤差逆伝播法

DAY3

- 学習の最適化
 - 勾配法の学習を最適化させる方法
 - 重みの初期値
 - 機械学習と純粋な最適化問題の差異
 - ニューラルネットワーク最適化の課題
 - 最適化戦略とメタアルゴリズム
 - 過学習と正則化

- バッチ正規化とその類似手法
- ドロップアウト
- 荷重減衰

DAY4

- ディープラーニングの様々なモデル
- 畳み込みニューラルネットワーク
 - CNN概要
 - 畳み込み層
 - プーリング層
 - im2col
- その他の話題
 - データ拡張
 - 構造出力
 - CNNで扱うデータの種類

DAY5

- 中間発表
- CNNの様々なモデル
 - 著名なCNNモデル
 - 物体検出タスクとCNN
 - セマンティックセグメンテーションタスクとCNN
- 自己符号化器

- 生成モデル
 - 生成モデルとは
 - 変分自己符号化器
 - 敵対的生成ネットワーク

DAY6

- 機械学習で扱うデータと典型的なタスク
 - 画像データ
 - 時系列データ
 - テキストデータ
 - データの権利
- 再帰型ニューラルネットワーク
 - 再帰型ニューラルネットワーク概要
 - シンプルなRNN
 - LSTM
 - GRU
 - RNNの発展モデル
 - その他の話題

DAY7

- 自然言語処理における深層学習
 - 自然言語処理と深層学習
 - 自然言語処理の基礎
 - word2vec

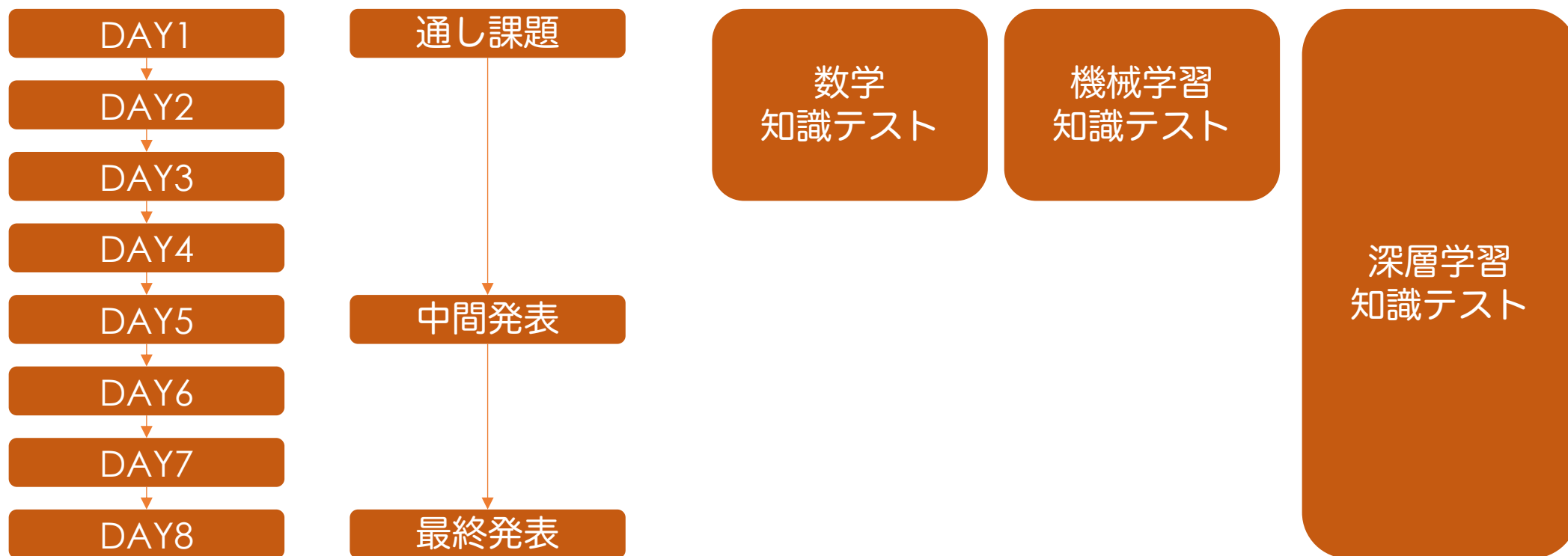
- 系列変換モデル
- アテンション
- トランスフォーマー
- 外部メモリを持つニューラルネットワーク
- その他の話題

DAY8

- 最終発表
- 強化学習
 - 強化学習の基礎1
 - 迷路問題
 - 強化学習の基礎2
 - 強化学習の各種手法
 - Deep Q-Network
 - カートポール問題
 - AlphaGO
 - 逆強化学習
 - 深層強化学習の実用面での課題
- 転移学習
- 軽量化技術
- 高速化技術

学習の流れ

- 本講座における学習の大まかな流れを以下に示します。
- 通し課題と知識テストは、各自のペースで取り組んで下さい。
- 中間発表と最終発表は、通し課題の進捗を発表して頂く機会です。参加は必須ではありませんが、講師からアドバイスを受けられる貴重な機会なので、ぜひご活用ください。



Any Questions ?

講座を通しての課題

講座を通しての課題

- 講座を通して、以下の課題に取り組んで頂きます。
 - 課題：手書きカタカナ「アイウエオカキクケコサシスセソ」の15文字を高い精度で識別できるモデルを構築せよ

課題で用いるデータについて

- 手書きカタカナ画像は、スキルアップAIが独自に制作したものです。
- 問題を簡単にするために、作者は数名にしました。
- 学習用データ(train)は、15文字それぞれ200枚ずつの合計3000枚です。これは講座開始時に配布します。
- テスト用データ(test)は、アイウエオはそれぞれ1300枚ずつ、カキクケコサシスセソはそれぞれ800枚ずつの合計14500枚です。これは講座終了後も受講者への配布は行いません。

課題の提出方法

- 学習済みモデルが完成したら、submit_katakana.ipynbを完成させ、学習済みモデル、自作コード、学習に使用したipynbなどを同じディレクトリに納めzip形式で圧縮し、識別精度算出ウェブサイトへ投稿してください。
- サーバーでテスト用データに対する正解率(Accuracy)を自動計算し、結果をメールでご連絡します。
- 投稿回数は3回/日を上限とします。
- テスト用データに対する正解率が講座修了要件の一つになります。その基準値は別途ご連絡します。
- 投稿できるzipファイルのサイズは、150MB以下です。それを超える場合は、学習済みモデルのファイルサイズを減らしてください。重みだけをpickleなどで出力するようにすれば、150MB以下に納まります。

課題で具体的に行うこと

1. 学習用データの生画像を加工し、計算しやすい形にする。(前処理)
2. モデルを構築する。
3. 学習用データにて、正解率を確認する。交差検証を行うのが理想。
4. 正解率が上がるように、学習方法を工夫したり、パラメータをチューニングしたりする。
5. 識別できない画像を確認し、その理由を考察する。
6. モデルが完成したら、提出形式に整えて識別精度算出ウェブサイトを投稿。
7. 結果をメールで受け取り、学習用データでの正解率と比較する。過学習が起きているようであれば、その対策を施す。
8. Arxiv.orgなどで参考になりそうな論文を探し、その知見をモデルに反映させる。
9. 講座終了までモデルの更新を繰り返す。

モデルの構築方針

- まずは、全結合型NNで組み上げてください。それができたらCNNのモデルをつくり、全結合型NNの結果とCNNの結果を比較して下さい。
- 講義の中でNNの各種関数を実装する方法を紹介します。それを参考に自分で実装して下さい。
- 識別精度算出ウェブサイトには、TensorFlowなどのフレームワークはインストールされていません。

中間発表

- 課題の成果を発表して頂きます。
- 1人当たりの時間は、**発表3分+質疑応答2分=5分**とします。
- 発表内容に入れるもの
 1. 実装の進捗状況
 2. モデルの改良点と正解率の変遷
 3. 学習用データでの正解率とテスト用データでの正解率の比較
 4. Arxiv.orgなどで見つけた論文とそこから得られた知見
 5. 今後の取り組み予定
- 事前に、課題をまとめたNotebookを講師にDMで渡してください。
- 発表時は、そのNotebookをプロジェクターに投影するようにします。時間が限られていますので、原則、個人PCでの発表は受け付けないようにしたいと思います。

最終発表

- 課題の成果を発表して頂きます。
- 1人当たりの時間は、**発表3分+質疑応答2分=5分**とします。
- 発表内容に入れるもの
 1. 実装の進捗状況
 2. モデルの改良点と正解率の変遷
 3. 学習用データでの正解率とテスト用データでの正解率の比較
 4. Arxiv.orgなどで見つけた論文とそこから得られた知見
 5. 課題を通しての感想
- 事前に、課題をまとめたNotebookを講師にDMで渡してください。
- 発表時は、そのNotebookをプロジェクターに投影するようにします。時間が限られていますので、原則、個人PCでの発表は受け付けないようにしたいと思います。

Any Questions ?

ディープラーニング基礎 前半

目次

1. ディープラーニング入門
2. パーセプトロン
3. ニューラルネットワーク
4. 活性化関数
5. 順伝播計算
6. 出力層の設計
7. 予測関数
8. バッチ処理
9. 損失関数

ディープラーニング入門

ディープラーニングとは

- ディープラーニングとは、深い層をもつニューラルネットワークを用いた学習手法のことである。
- どこからがディープなニューラルネットワークで、どこまでだったらディープでないニューラルネットワーク、という明確な定義はない。
- ニューラルネットワーク自体は、1950年ごろから研究されていたが、多層化されたものは学習がうまくできなかった。
- 2010年頃から、ディープニューラルネットワークが成果を出し始めた理由としては、手法の研究が進んだこと、計算機の発達、豊富なデータを容易に入手できるようになったこと、などが挙げられる。

ディープラーニングまわりの用語

ニューラルネットワーク、パーセプトロン

最適化手法、最急降下法、勾配降下法、確率的勾配降下法、学習率

誤差逆伝播法

活性化関数

損失関数

バッチ学習、ミニバッチ学習

過学習、正則化

畳み込み、プーリング

学習済みモデル、再学習(ファインチューニング)、転移学習(トランスファーラーニング)

音声認識

画像認識

生成モデル

強化学習

異常検知

時系列解析

機械学習分野の分類とディープラーニング

- ディープラーニングは、主に回帰問題や分類問題で利用されるが、教師なし学習、異常検知、強化学習でも利用される。

教師あり

回帰
分類

教師なし

特徴量抽出
次元削減

中間的手法

異常検知
強化学習

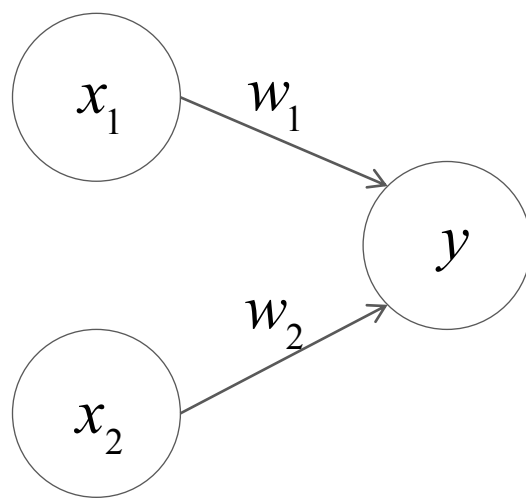
ディープラーニングのフレームワーク有名どころ

フレームワーク	公開年	開発者	言語
Caffe	2013	Berkeley Vision and Learning Center	C++, Python
Torch7	2011	University of New York	C, Lua
Theano	2010	University of Montreal	Python
Pylearn2	2013	University of Montreal	Python
CNTK	2016	Microsoft	C++
TensorFlow	2015	Google	C++, Python
Chainer	2015	PFN	Python
Pytorch	2016	PyTorch is currently maintained by Adam Paszke, Sam Gross, Soumith Chintala and Gregory Chanan	Python
NNabla	2017	Sony	Python

パーセプトロン

パーセプトロンとは

- パーセプトロンとは、複数の信号を受け取り、1つの信号を出力するアルゴリズムのこと。
- 出力されるものは、信号を流す(1)か流さないか(0)の2値。
- 2つの信号を受け取るパーセプトロンの例を以下に示す。
- 図の○はノードと呼ばれ、 w_1, w_2 は重みと呼ばれる。
- ノード y では、送られてきた信号の総和が計算され、その総和が限界値をこえた場合にのみ1を出力する。



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

ANDゲート

- パーセプトロンを使って簡単な問題を考えてみる。
- 下表のような入出力対応をするパーセプトロンを実現するには、 w_1, w_2, θ をどのように決めればいいか？
- この入出力対応を実現する回路のことをANDゲートと呼ぶ。

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

NANDゲート

- 下表のような入出力対応をするパーセプトロンを実現するには、 w_1, w_2, θ をどのように決めればいいか？
- この入出力対応を実現する回路のことをNANDゲートと呼ぶ。

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

ORゲート

- 下表のような入出力対応をするパーセプトロンを実現するには、 w_1, w_2, θ をどのように決めればいいか？
- この入出力対応を実現する回路のことをORゲートと呼ぶ。

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

重み w_1, w_2 と閾値 θ

- ここでは w_1, w_2, θ を人が考えた。
- ディープラーニングなどの機械学習では、この作業をコンピュータに自動で行わせる。
- この作業のことを学習と呼ぶ。

[演習] ANDゲート、NANDゲート、ORゲートの実装

- 1_1_perceptron_trainee.ipynb
 - NANDゲートを実装しましょう。
 - ORゲートを実装しましょう。
 - それぞれ、シンプルな実装方法、またはNNでの利用を意識した実装方法のどちらかで実装しましょう。

シンプルな実装を行うときの式

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

NNでの利用を意識した実装を行うときの式

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

ここでのbはバイアス。wと一緒にして重みと呼ばれることもある。

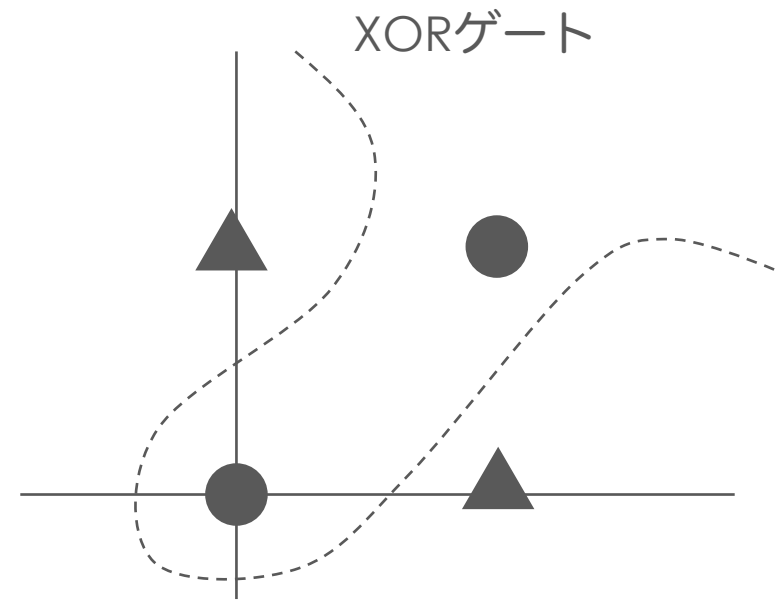
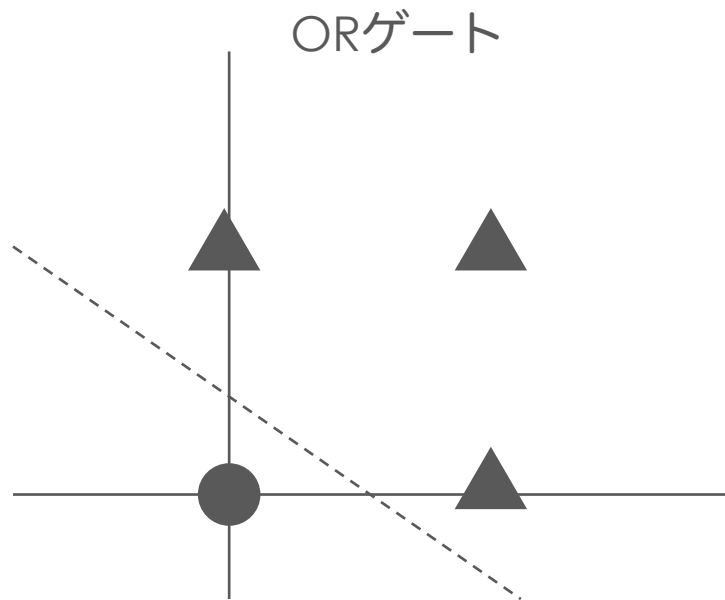
XORゲート

- 下表のような入出力対応をするパーセプトロンを実現するには、 w_1, w_2, θ をどのように決めればいいか？
- この入出力対応を実現する回路のことをXORゲートと呼ぶ。

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

XORゲート

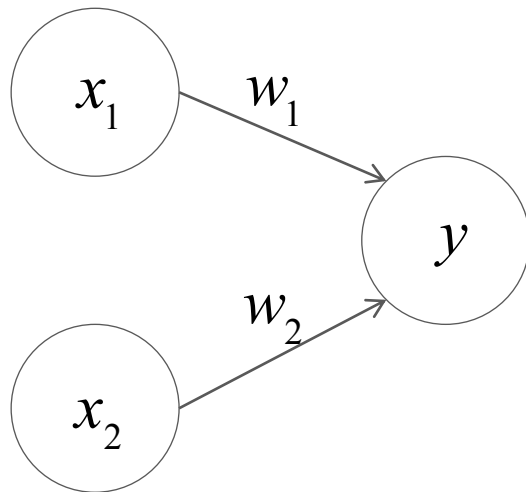
- XORゲートを実現できる w_1, w_2, θ は存在しない。
- その理由を以下の図で考える。パーセプトロンの目的は、図で示すと、0(●)と1(▲)を直線で分離する直線を見つけること。
- ORゲートは、その直線が存在するが、XORゲートは非線形な線でないと分離できない。
- XORゲートを実現するにはどうすればいいか？



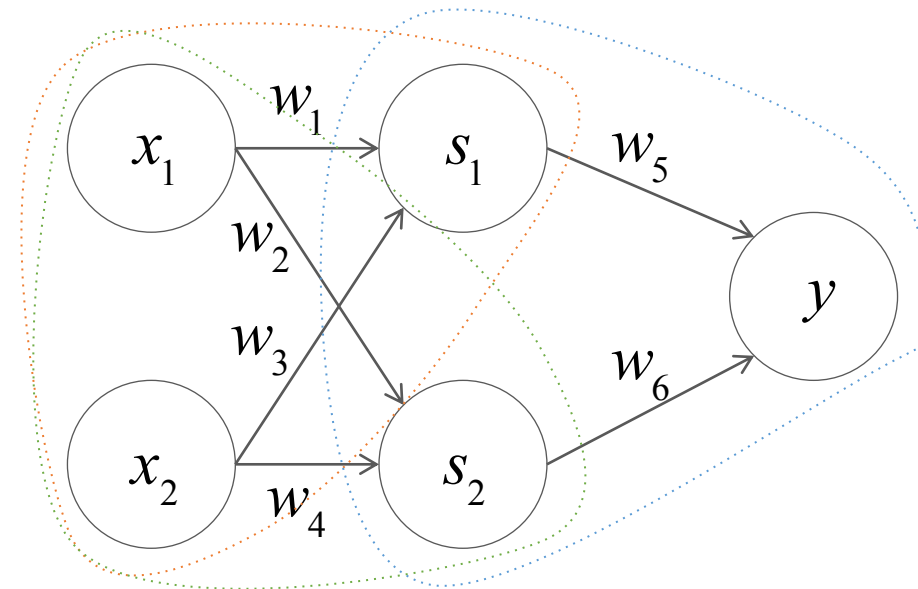
XORゲート

- XORゲートを実現するには、複数のゲートを組み合わせるというのも1つの方法。
- 3つのゲートを組み合わせるとXORが実現する。3つをどのように組み合わせればいいのか？
- 3つのゲートを組み合わせたモデルは、2層のパーセプトロンと呼ばれる。

1つのゲート



3つのゲート

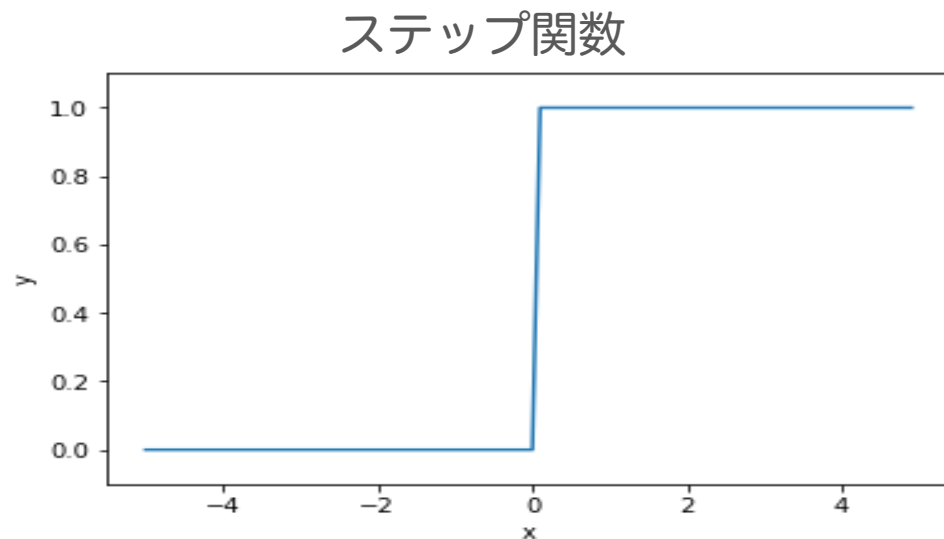


[演習] XORゲートの実装

- 1_1_perceptron_trainee.ipynb
 - ANDゲート、NANDゲート、ORゲートを組み合わせてXORゲートを完成させましょう。

パーセプトロンと活性化関数

- パーセプトロンでは、総和が閾値を超えていれば1、閾値以下であれば0を出力していた。
- この閾値判定の仕組みには、ステップ関数という名前が付いている。
- このステップ関数は活性化関数の1つ。活性化関数の詳細は後述する。
- パーセプトロンでは、このステップ関数を用いているため、多層化することによって入出力の非線形な関係を表現することができる。

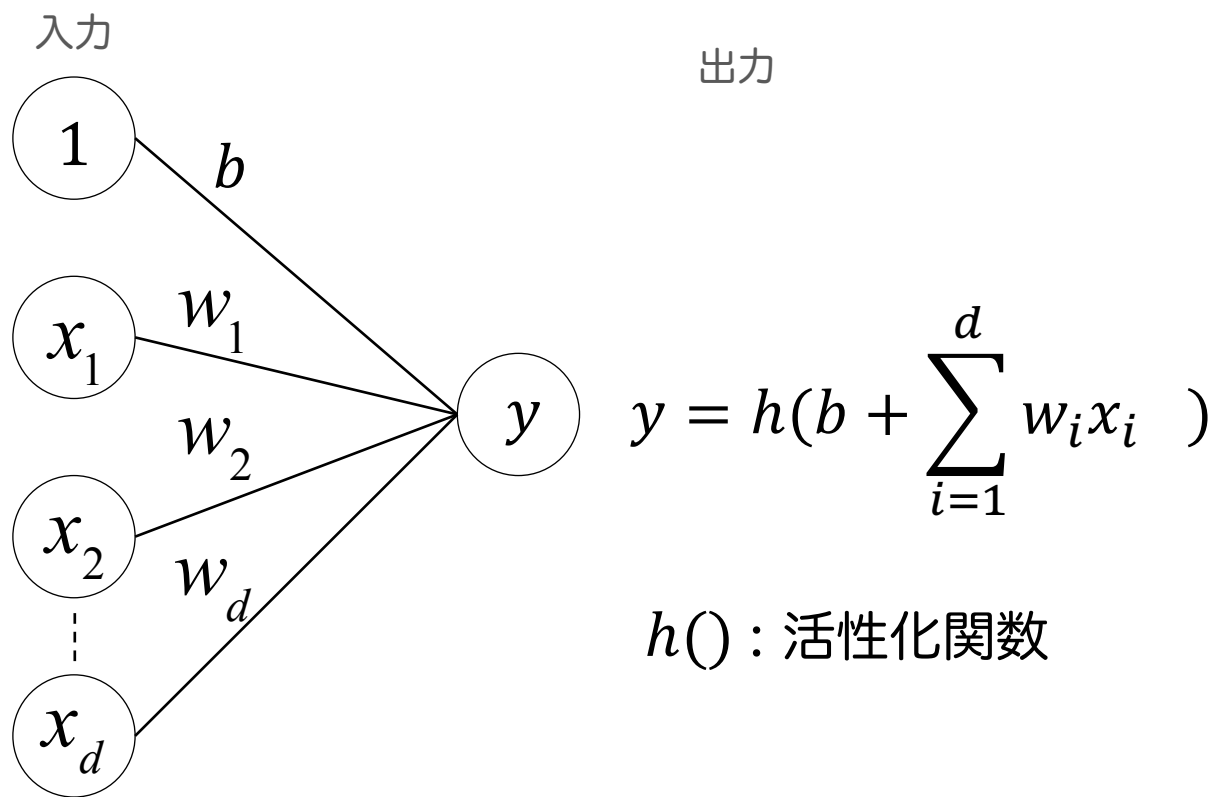


Any Questions ?

ニューラルネットワーク

パーセプトロン

- パーセプトロンとは、入力層と出力層だけで構成されたネットワークモデルのこと。
- 入力に重みをかけた総和を計算し、その結果を活性化関数に通したものが出力となる。活性化関数にはステップ関数が用いられる。



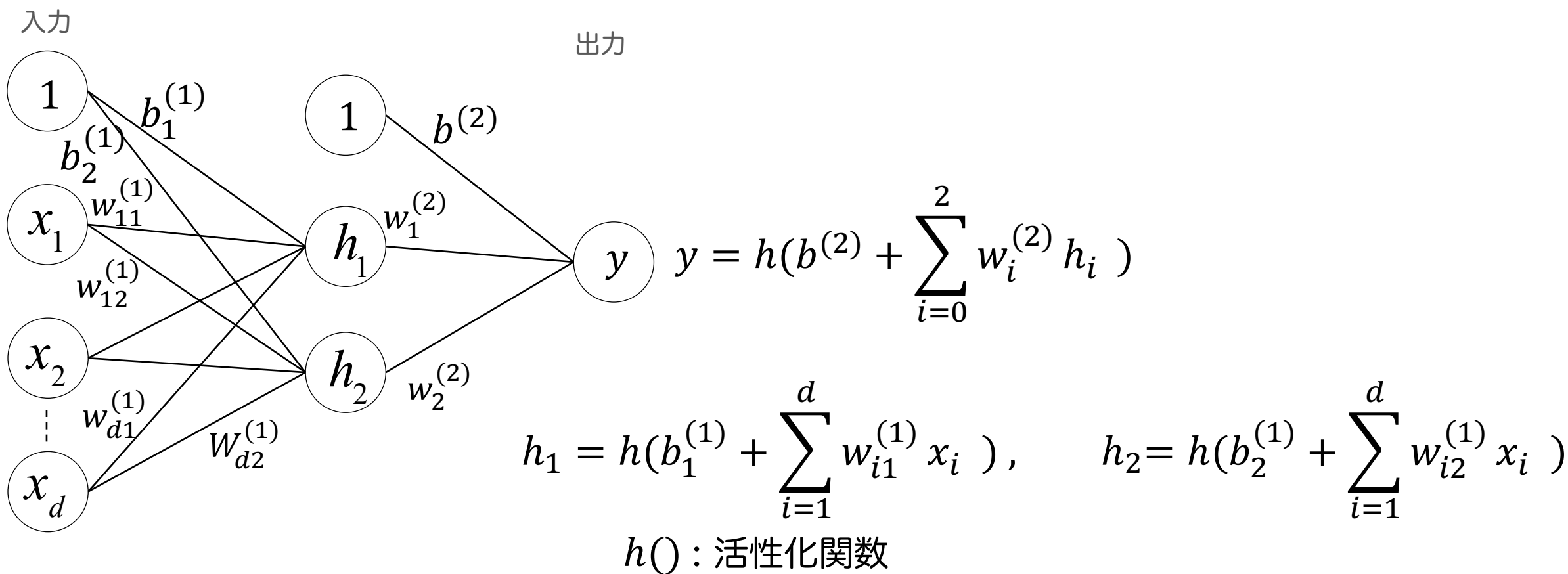
多層パーセプトロン

重みの記号

$w_{12}^{(1)}$ 第1層目の重み

前層の1つ目のノード 次層の2つ目のノード

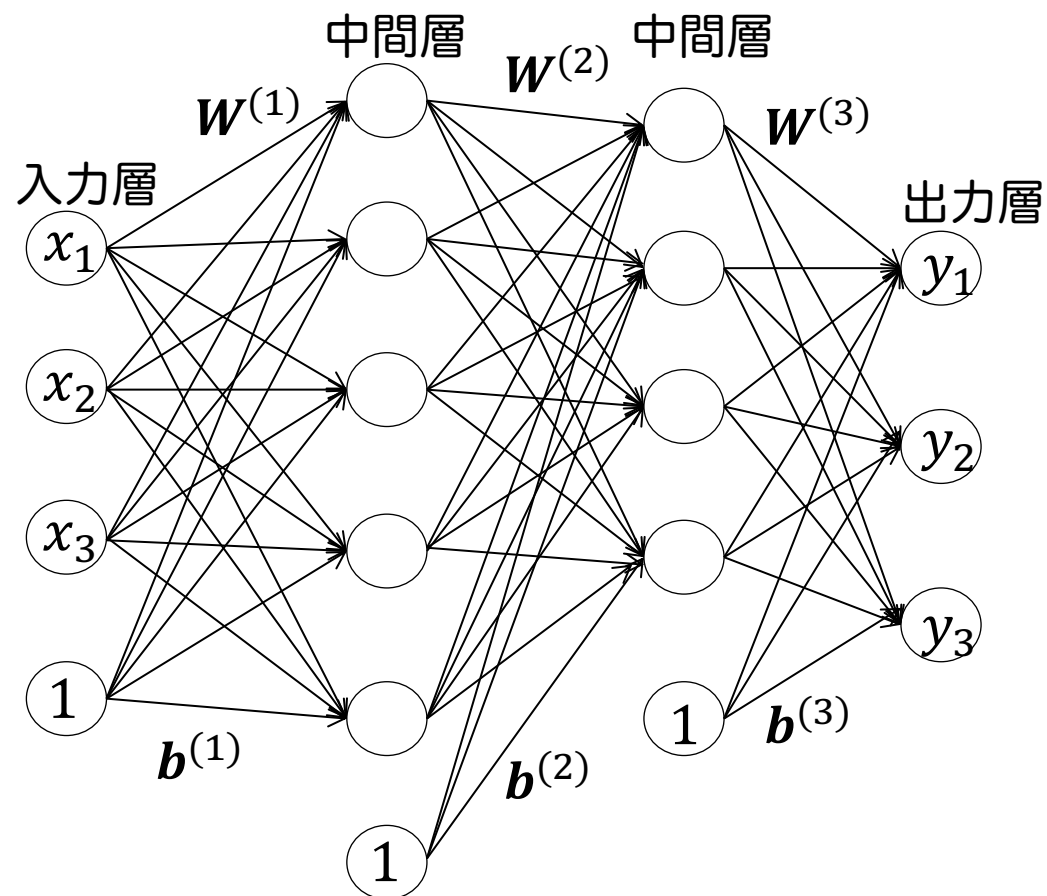
- 多層パーセプトロンとは、入力層と出力層の間に中間層(隠れ層)があるパーセプトロンのこと。



ニューラルネットワーク

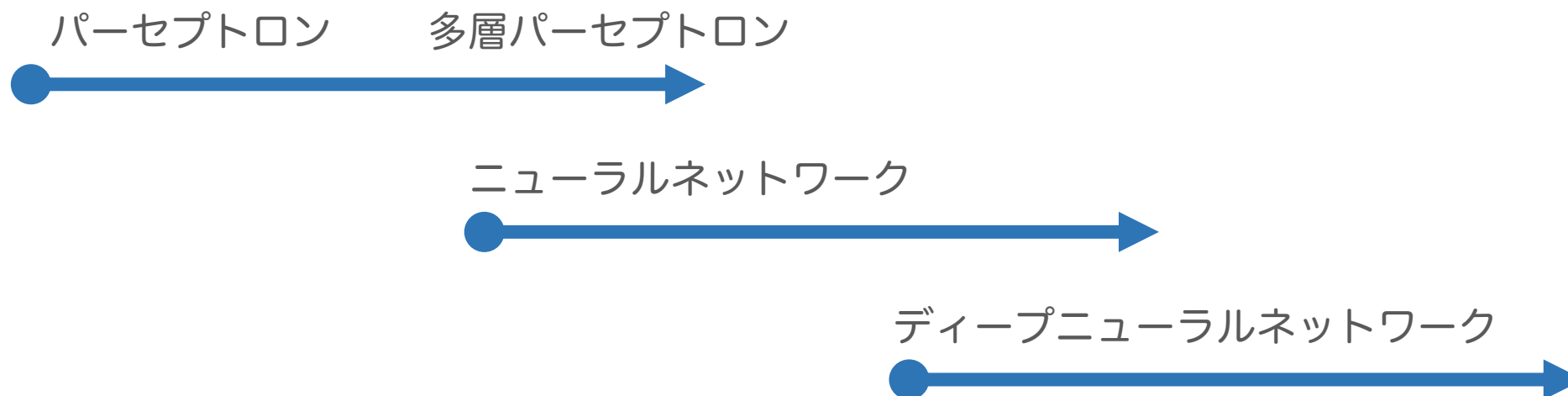
- 多層パーセプトロンとほぼ同じ意味。

ニューラルネットワークの例



パーセプトロンからディープニューラルネットワークまで

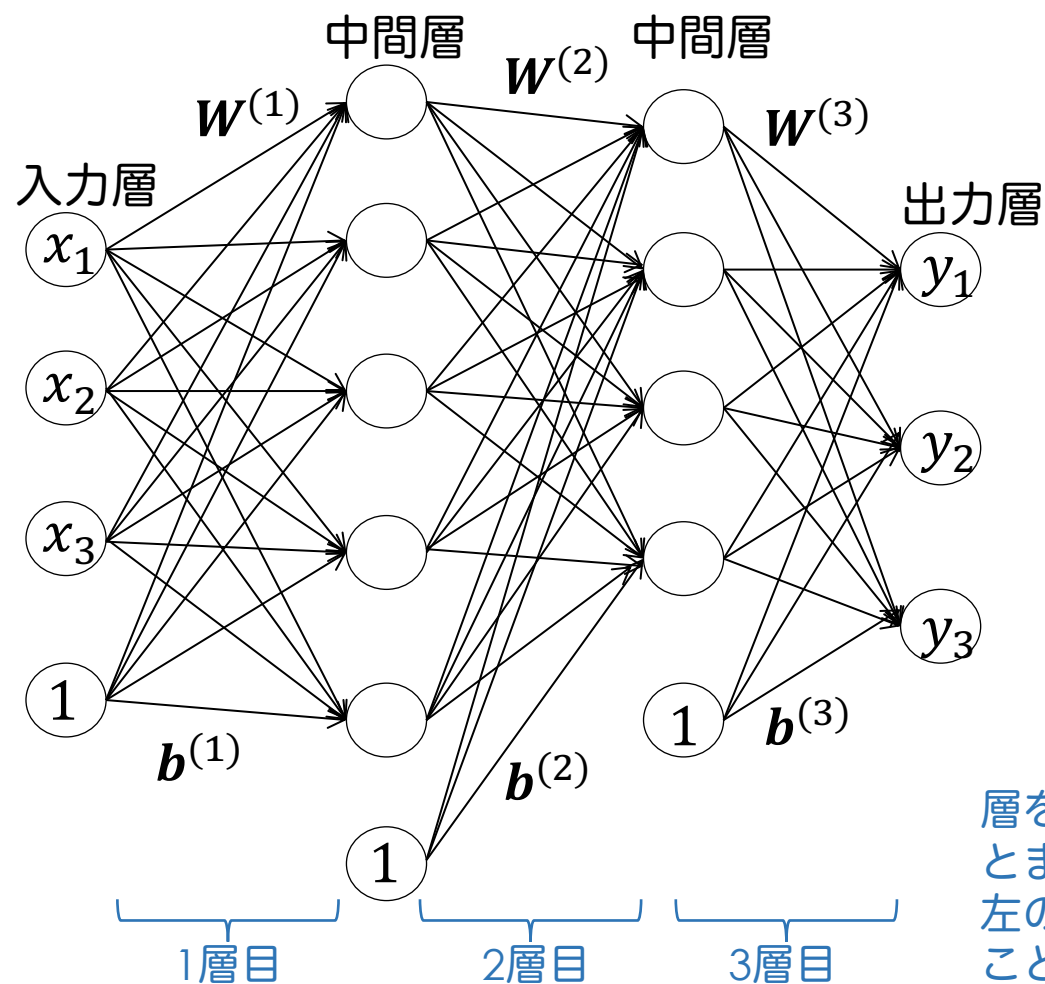
- パーセプトロン、ニューラルネットワーク、ディープニューラルネットワークの言葉の境界はあいまい。



右にいくほどモデルは複雑になっていく

層という言葉の使い方

ノードの縦のまとまりを層と呼ぶ



層を数えるときは、層間の重みをひとまとまりに扱えた方が便利なので、左のように、1層目、2層目と数えることが多い。

万能近似定理

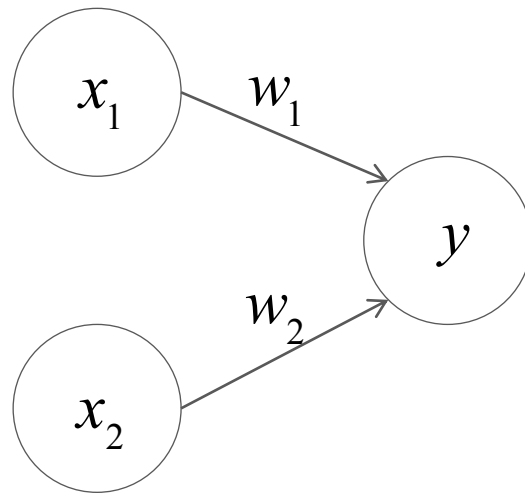
- 万能近似定理(universal approximation theorem)は、ネットワークが十分な数の隠れユニットを持つ場合、線形出力層とシグモイド関数のような「押しつぶす」活性化関数をもつ隠れ層が少なくとも1つ含まれる順伝播型ネットワークは、どんなボレル可測関数でも任意の精度で近似できる、と述べている。
 - 『深層学習』(松尾豊監修、KADOKAWA) 6.4.1節参照
 - \mathbb{R}^n の有界で閉じた部分集合上の任意の連続関数は、ボレル可測関数である
- つまり、ノードを増やしていくと、ニューラルネットワークの表現力はどんどん上がっていき、学習データをほぼ完全に説明できるニューラルネットワークが実現できる、ということ。
 - 学習データをほぼ完全に説明できるモデルが実現できたとしても、そのモデルの汎化性能が必ずしも高いわけではないことに注意。

Any Questions ?

活性化関数

パーセプトロンの復習

- パーセプトロンとは、入力の重み付け和が閾値 θ を超えると y が1になるモデルであった。

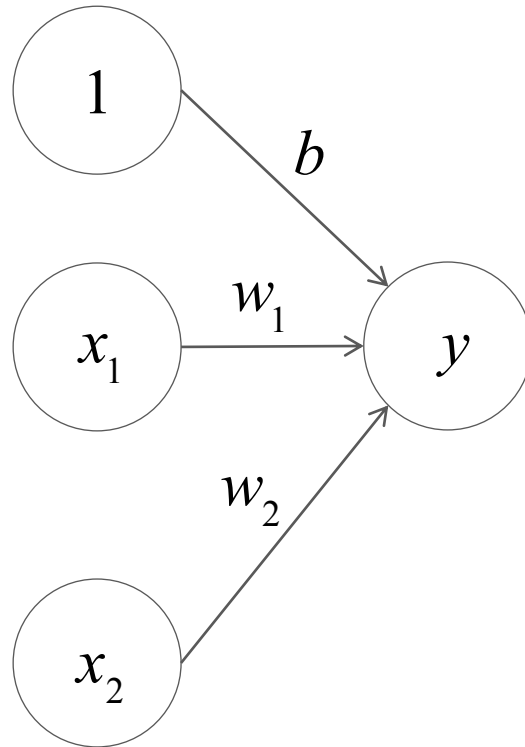


シンプルな実装を行うときの式

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

パーセプトロンの復習

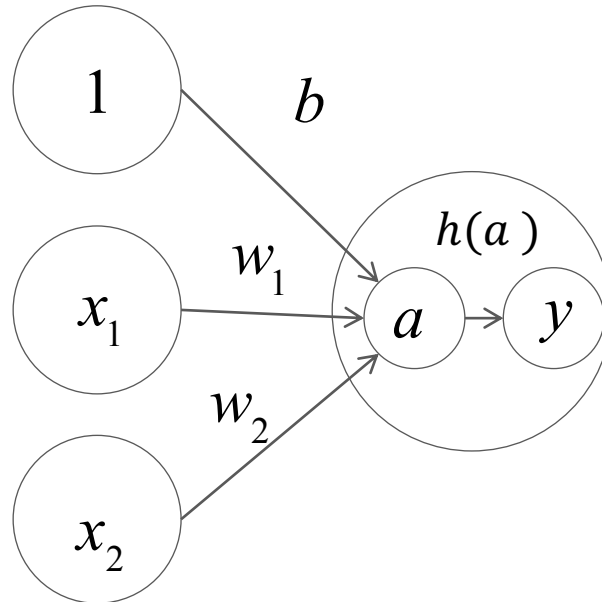
- 閾値 θ をバイアス b で表現することもあった。



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

活性化関数とは

- 「閾値を超えると1を出力する」という機能を明示的に図示すると以下のようなになる。
- この $h(a)$ のことを活性化関数と呼ぶ。

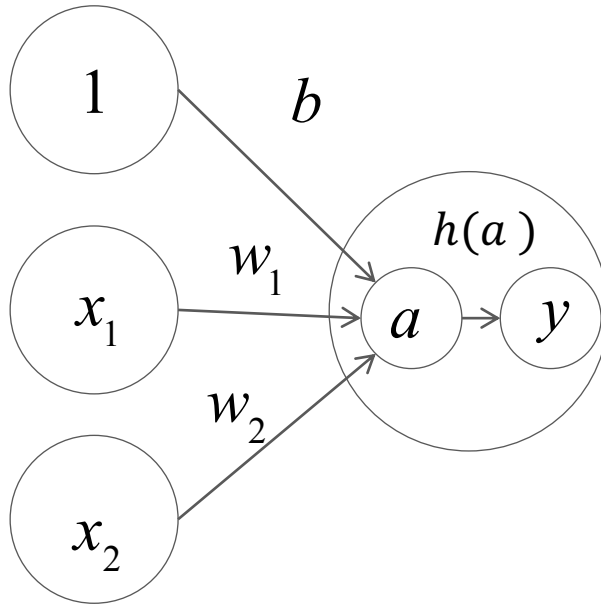


$$y = h(b + w_1x_1 + w_2x_2)$$

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

活性化関数の意義

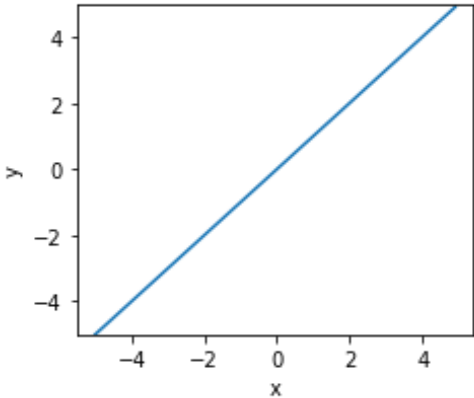
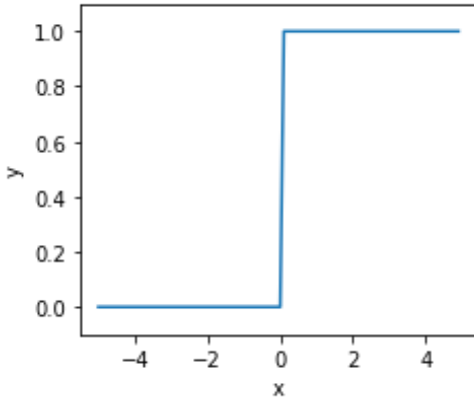
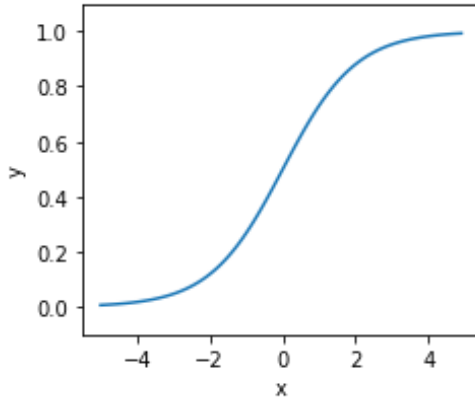
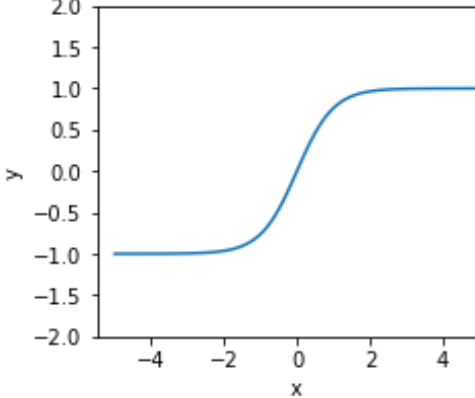
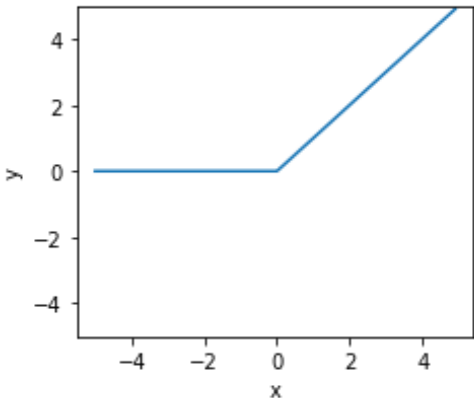
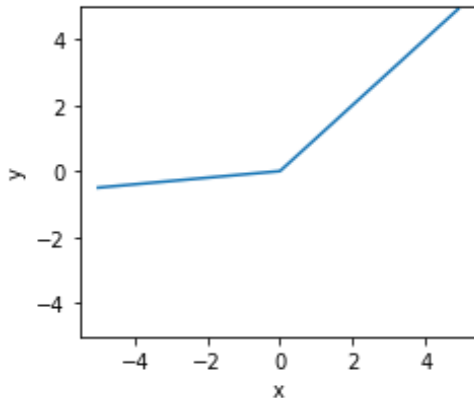
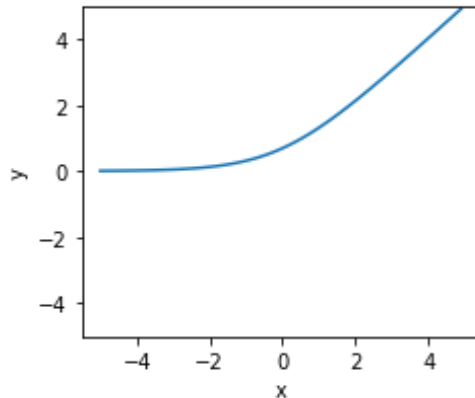
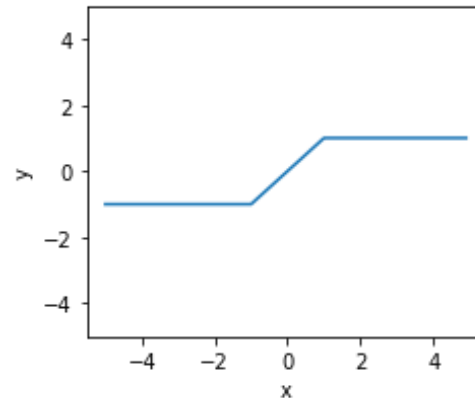
- 活性化関数 $h(a)$ が無いと、単なる線形結合となり、線形回帰とほぼ同じ計算モデルとなる。
- 線形回帰でモデリングできるのであれば、ニューラルネットワークでモデリングする必要性がなくなる。
- 活性化関数 $h(a)$ があると、 x と y の関係が非線形化され、ニューラルネットワークでモデリングする意義が出てくる。



$$y = h(b + w_1x_1 + w_2x_2)$$

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

活性化関数の種類

恒等写像関数	ステップ関数	シグモイド関数	tanh関数
			
ReLU関数	LeakyReLU関数	Softplus関数	Hardtanh関数
			

[演習] 活性化関数の実装

- 1_2_activation_function_trainee.ipynb
 - 活性化関数を実装しましょう。

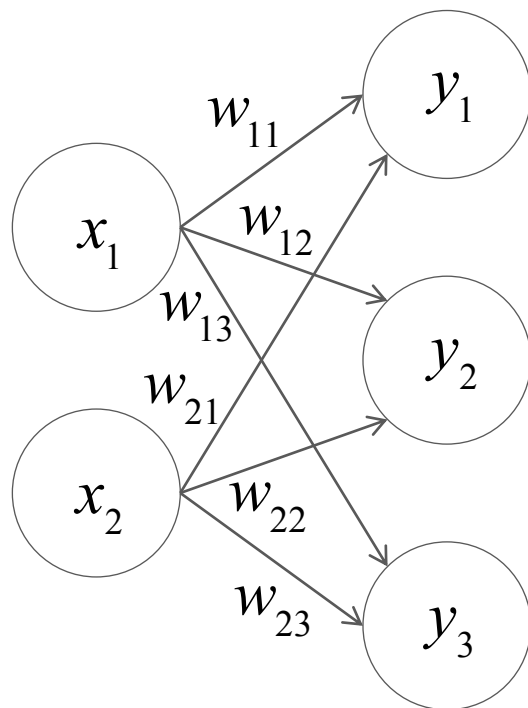
順伝播計算

順伝播計算とは

- 入力層から出力層に向かって、データを流していくことを順伝播計算という。
- これとは逆に、出力層から入力層に向かって、データを流していくことを逆伝播計算という。逆伝播計算の必要性和計算方法については、DAY2で説明する。

順伝播計算の効率化

- 順伝播計算では、ある層への入力の重み付き和を計算するが、この計算に行列を用いると計算が楽になり、数式も簡潔に表現できる。なお、逆伝播計算でも同様の行列計算を用いる。
- 入力層と出力層だけの単純なニューラルネットワークの例を以下に示す。(活性化関数は省略)



行列の計算

$$\begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} = (x_1 w_{11} + x_2 w_{21} \quad x_1 w_{12} + x_2 w_{22} \quad x_1 w_{13} + x_2 w_{23}) = (y_1 \quad y_2 \quad y_3)$$

行列の計算の簡潔な表記

$$\mathbf{XW} = \mathbf{Y}$$

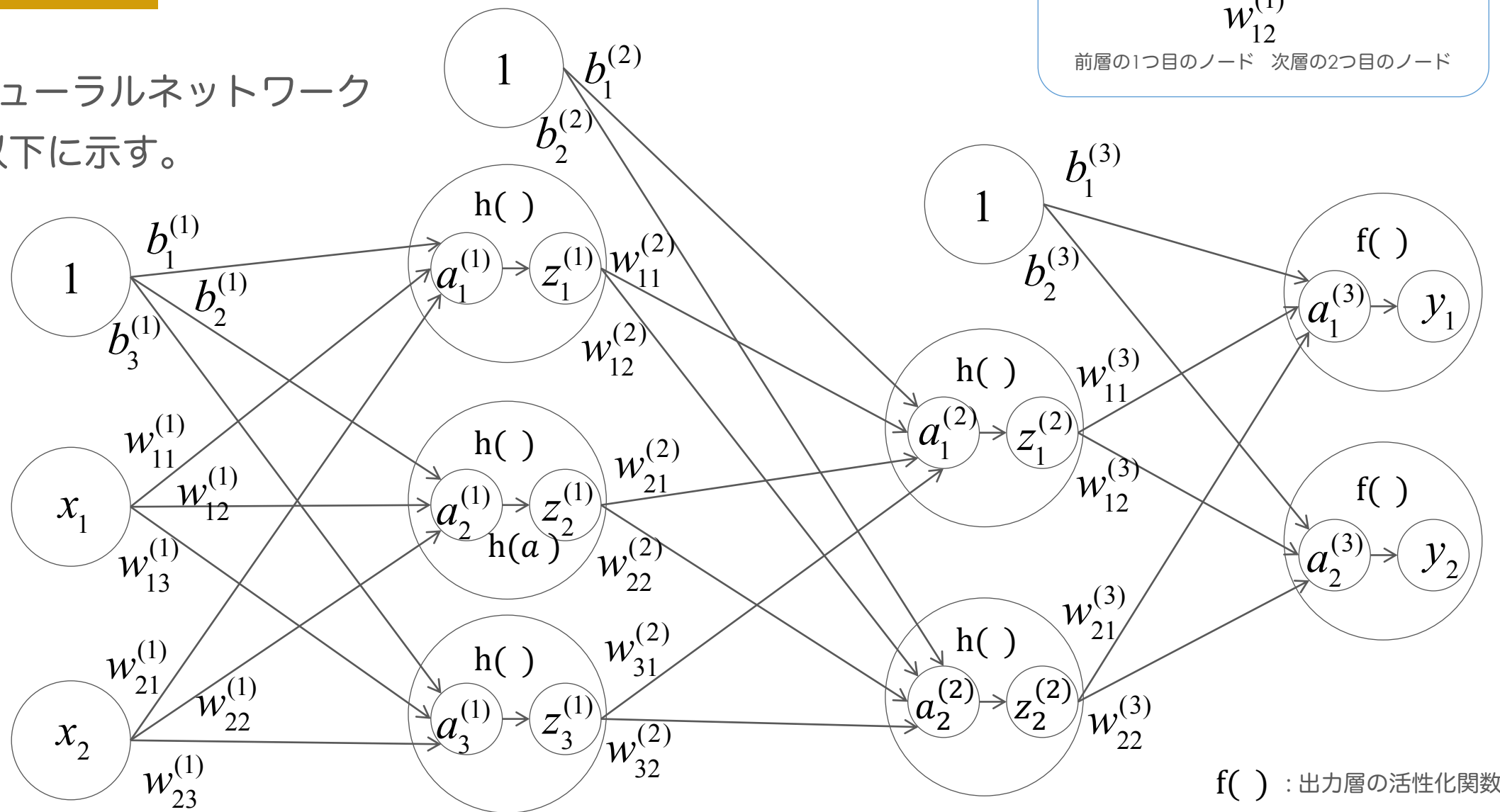
※ここで、 \mathbf{x}, \mathbf{y} はベクトルではなく行列であることに注意

[演習] 順伝播計算の実装

- 1_3_forward.ipynb
 - 1層のニューラルネットワークの実装を確認しましょう。
 - 3層のニューラルネットワークの実装を確認しましょう。
 - 参考として、3層ニューラルネットワークの図を次頁に掲載しています。

3層のニューラルネットワークの例

- 3層のニューラルネットワークの例を以下に示す。



出力層の設計

出力層の設計

- 回帰問題では恒等関数を用いる。
- 2クラス分類問題では、シグモイド関数を用いる。2クラス分類問題を多クラス分類問題とみなし、ソフトマックス関数を用いることもできるが、最適化するパラメータが多くなってしまう。
 - 例えば、2クラス分類問題において、シグモイド関数を用いる場合、出力層は1ノードで済むが、ソフトマックス関数を用いる場合は出力層が2ノードになる。これより、出力層の直前の重みの数が2倍になる。
- 多クラス分類問題では、ソフトマックス関数を用いる。

ソフトマックス関数

- ソフトマックス関数は、複数の入力を正規化し、合計値が1になるようにする関数
- 正規化する前に、指数関数(exp)を計算している。

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

y_k :出力層のノード番号 k に対応するソフトマックスの出力値

K :出力層のノード数

k :出力層のノード番号

[演習] ソフトマックス関数の実装

- 1_4_softmax_trainee.ipynb
 - ソフトマックス関数を実装しましょう。
 - 実装時は、オーバーフローに注意しましょう。
 - ソフトマックス関数の性質を確認しましょう。

[演習] 線形回帰モデルとNNの比較

- 1_5_linear_regression_trainee.ipynb
 - 住宅価格予測問題を線形回帰で解いてみましょう。
計算にはscikit-learnを用います。
 - 住宅価格予測問題を線形のNNで解いてみましょう。
計算にはtensorflow+kerasを用います。
 - 両者のモデルを比較しましょう。

Any Questions ?

予測関数

「学習」と「予測」

- 本講座では、学習用データを用いて最適なパラメータ(W, b など)を求めることを「学習」、あるパラメータ(W, b など)を用いて何らかの予測やクラス分類を行うことを「予測」と呼ぶことにする。

[演習] 予測関数を実装する

- 1_6_mnist.ipynb
 - MNISTデータの中身を確認しましょう。
- 1_7_predict_trainee.ipynb
 - 予測関数を実装しましょう。

Any Questions ?

バッチ処理

バッチ処理

- バッチ処理とは、入力データを束ねる(バッチ)ことによって、計算を効率的に行う方法のこと。
- Numpyなどの数値演算ライブラリは、大きな配列の計算を効率的に処理できるように実装されているため、小さな配列を何度も計算するよりも小さな配列を束ねて一度に計算した方が計算が速くなる。
- 束ねるデータの数のことをバッチサイズと呼ぶ。
- バッチサイズを大きくすると、それに応じてメモリ使用量が増える。 GPUを使う場合は、バッチサイズの上限は、GPUメモリ量で決まることが多い。

[演習] バッチ処理を実装する

- 1_8_batch_trainee.ipynb
 - バッチ処理によって予測を行う部分を実装しましょう。
 - MNISTの学習済みパラメータを使って、実装が正しいことを確認しましょう。

Any Questions ?

損失関数

損失関数

- 損失関数は、ニューラルネットワークの「悪さ」を表す指標。
- 現在の重みが教師データに対してどれくらい適合していないかを表す。
- 回帰問題であれば、2乗和誤差がよく用いられる。
- 分類問題であれば、クロスエントロピー誤差がよく用いられる。

2乗和誤差関数

2乗和誤差

$$L = \frac{1}{2} \sum_{k=1}^K (y_k - t_k)^2$$

L :損失関数

K :出力層のノード数

k :出力層のノード番号

y_k :ノード k の出力値

t_k :ノード k の正解値

2乗和誤差(バッチ対応版)

$$L = \frac{1}{N} \sum_n \left(\frac{1}{2} \sum_k (y_{nk} - t_{nk})^2 \right) = \frac{1}{2N} \sum_n \sum_k (y_{nk} - t_{nk})^2$$

L :損失関数

N :データ数

n :データ番号

K :出力層のノード数

k :出力層のノード番号

y_{nk} :データ n のノード k の出力値

t_{nk} :データ n のノード k の正解値

クロスエントロピー誤差関数

クロスエントロピー誤差

$$L = - \sum_k^K t_k \log y_k$$

L : 損失関数

K : 出力層のノード数

k : 出力層のノード番号

y_k : ノード k の出力値 (通常は、0 と 1 の間を取る値)

t_k : ノード k の正解値 (通常は、0 or 1. つまり t は onehot ベクトル)

t は通常、ワンホットベクトル (one-hot vector) である。つまり、クロスエントロピーの計算では、 $t_k = 1$ に対応する y_k の値しか利用されないことになる。 $t_k = 0$ に対応する y_k は無視される。

クロスエントロピー誤差(バッチ対応版)

$$L = \frac{1}{N} \sum_n^N \left(- \sum_k^K t_{nk} \log y_{nk} \right) = - \frac{1}{N} \sum_n^N \sum_k^K t_{nk} \log y_{nk}$$

L : 損失関数

N : データ数

n : データ番号

K : 出力層のノード数

k : 出力層のノード番号

y_{nk} : データ n のノード k の出力値 (通常は、0 と 1 の間を取る値)

t_{nk} : データ n のノード k の正解値 (通常は、0 or 1. つまり t は onehot ベクトル)

ワンホットベクトルとは

- ワンホットベクトル(one-hot vector)とは、ある1つの要素だけが1であり、その他の要素が0であるベクトルのこと。
- 例えば、0~9の数字のうちどれかであることを表現するワンホットベクトルの場合、
 - $y = \{0, 0, 1, 0, 0, 0, 0, 0, 0, 0\} \rightarrow 2$ を意味する
 - $y = \{0, 0, 0, 0, 1, 0, 0, 0, 0, 0\} \rightarrow 4$ を意味する
 - $y = \{0, 0, 0, 0, 0, 0, 0, 0, 1, 0\} \rightarrow 8$ を意味する

クロスエントロピーとは

- クロスエントロピーとは、2つの確率分布の異なりを表す指標。
- 損失関数に用いられる場合は、正解の分布と予測結果の分布の異なり具合を表す。
- 正解の分布と予測結果の分布が異なるほど、クロスエントロピーの値は大きくなる。
- 予測結果の情報量を正解分布で期待値をとったものと解釈することもできる。

[演習] 損失関数を実装する

- 1_9_loss_function_trainee.ipynb
 - 損失関数を実装しましょう。

Any Questions ?

[グループワーク] 活性化関数の調査

- 2~3名のグループに分かれて、本スライドで紹介した8つの活性化関数以外の活性化関数を探してみましょう。(15分)
- 8つの活性化関数+見つけた活性化関数について、それぞれの特徴を話し合しましょう。(30分)
- 最後に、グループごとに発表していただきます。(15分)

講座の時間が余ったら

- 今回の復習をします。
- 次回の予習をします。

Appendix

情報量とは

情報量とは、確率変数 x の値を得た際の驚きの度合いを表す指標。起きそうもない事象が起きたと知ったら、いつでも起きそうな事象が起きたことを知るよりも多くの情報を得たといえる。

$$h(x) = -\log_2 p(x)$$

$h(x)$: 情報量

x : 確率変数

$p(x)$: x となる確率

ただし、

$$\sum p(x) = 1$$

$$\max(p(x)) \leq 1$$

$p(x) = 0$ のときは $\log_2(x) = 0$ とする

[例]

例えば、東京に雨が降ったという情報よりも、砂漠に雨が降った(確率が低い事象)という情報の方が驚くはずである。

ある日に、東京に雨が降る確率 $p(x)$ を0.3とすれば、その情報量は、

$$h(x) = -\log_2 0.3 = 1.737$$

となり、ある日に、砂漠に雨が降る確率 $p(x)$ を0.01とすれば、その情報量は、

$$h(x) = -\log_2 0.01 = 6.644$$

となる。

エントロピー

エントロピー(平均情報量)とは

エントロピーとは、情報量を平均化したものであり、確率変数 x のばらつき具合を表す。

エントロピーは下式により求められる。

$$E = -\sum p(x) \log_2 p(x)$$

E : エントロピー

x : 確率変数

$p(x)$: x となる確率

ここで、 E は分布 $-\log_2 p(x)$ の期待値に相当する。

[例]

例えば、ある日の東京の天気確率が、 $p(\text{晴れ})=0.5$ 、 $p(\text{雨})=0.3$ 、 $p(\text{曇り})=0.2$ とすると、そのエントロピーは、

$$E = -\sum p(x) \log_2 p(x) = 1.485$$

となり、ある日の砂漠の天気確率が、 $p(\text{晴れ})=0.9$ 、 $p(\text{雨})=0.01$ 、 $p(\text{曇り})=0.09$ とすると、そのエントロピーは、

$$E = -\sum p(x) \log_2 p(x) = 0.516$$

となる。

東京の天気は、確率変数 x がばらついており、砂漠の天気に比べ予測が難しいと言える。