

ニューラルネットワークの入門

単純な演算から画像認識まで —

初めてまして

- メネンデス・フランシスコ (Menendez Francisco)
- アルゼンチン出身・2003年日本に移民
- 最近はまっていること
 - DEATH STRANDING
- 岩手大学大学院卒業（博士）
- 単純な履歴
 - 青森大学（3年間・助教）
 - プレミアムエージェンシー（2年間・ゲーム開発）
 - ケイブ（2年間・ゲーム開発）
 - サイバーコア（4年間・画像処理、AI開発）
- 現在
 - イロジック・ゲームズ（一応、すべて）
 - 画像処理、AIについてコンサルタント

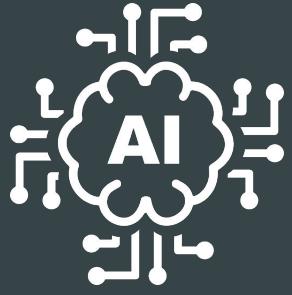


- AIとは
- ニューラルネットワークの仕組み
 - 世界で最も単純な課題：ANDとORを再現
- ニューラルネットワークの学習
 - 勾配関数について
 - 学習率とは
- Keras：ニューラルネットワークの開発フレームワーク
- 多層（ディープ）ネットワーク
- 画像認識①
- 画像認識②
- 回帰
- おまけ：AIの落とし穴（※時間があれば…）
- まとめ

本スライド、およびソースコード

<https://github.com/Arkaid/ai-seminar>

menendez.f@illogic.games



AIとは

AI : Artificial Intelligence

人工知能

人工知能: 人間を真似する仕掛け

・例: 自然言語処理

- ・文章を分解し、意味を把握し、回答できる
- ・Siri, Alexa(一部)

・例: Expert Systems

- ・巨大な if-then のルールで判断
- ・不具合の診断
- ・専門家の知識を電子化

・ホワイトボックス: 人がルールを設定し、処理の流れを解析できる

機械学習: データから学ぶ

・例: 線形回帰

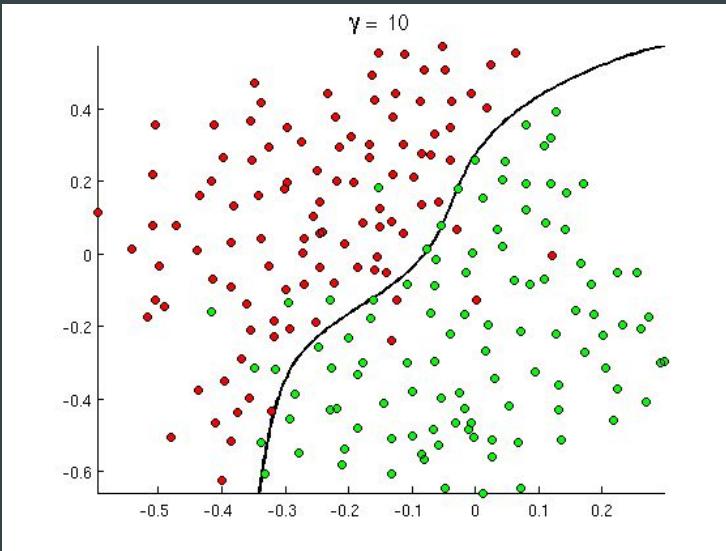
- ・ルールを設定せず、十分にデータがあれば、アルゴリズムがルールを学ぶ

・グレー～ブラックボックス: システムは複雑ほど、処理の流れが不透明になってしまふ

ニューラルネットワーク

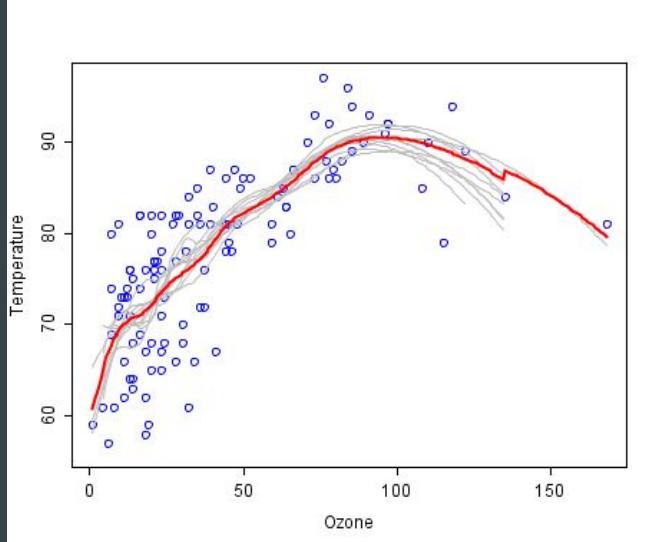
AIとは ニューラルネットワークが対応できるのは

分類



- ・データを定めた「クラス」（種類）に分割
- ・出力は離散 ($0, 1, 2 \cdots n$)

回帰



- ・入力と出力の相関を求める
- ・出力は連続（実数）

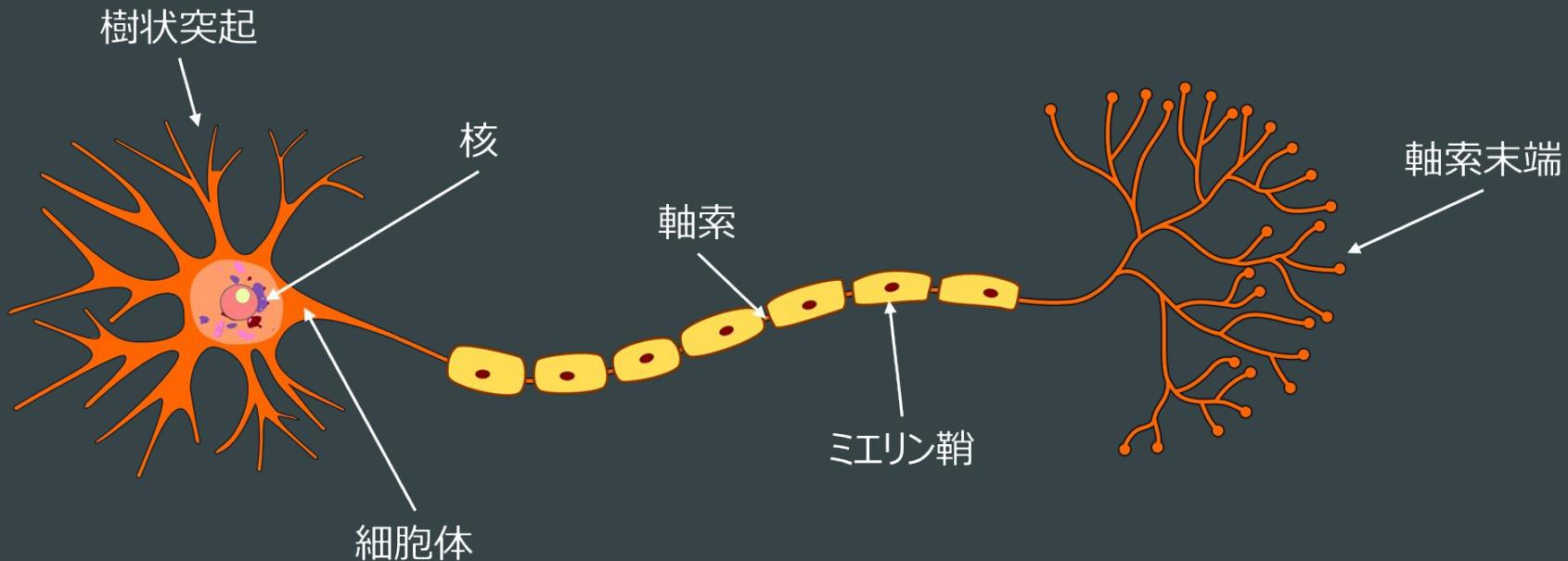
- ・特徴：データを記述する変数
- ・特徴は数次元でもOK



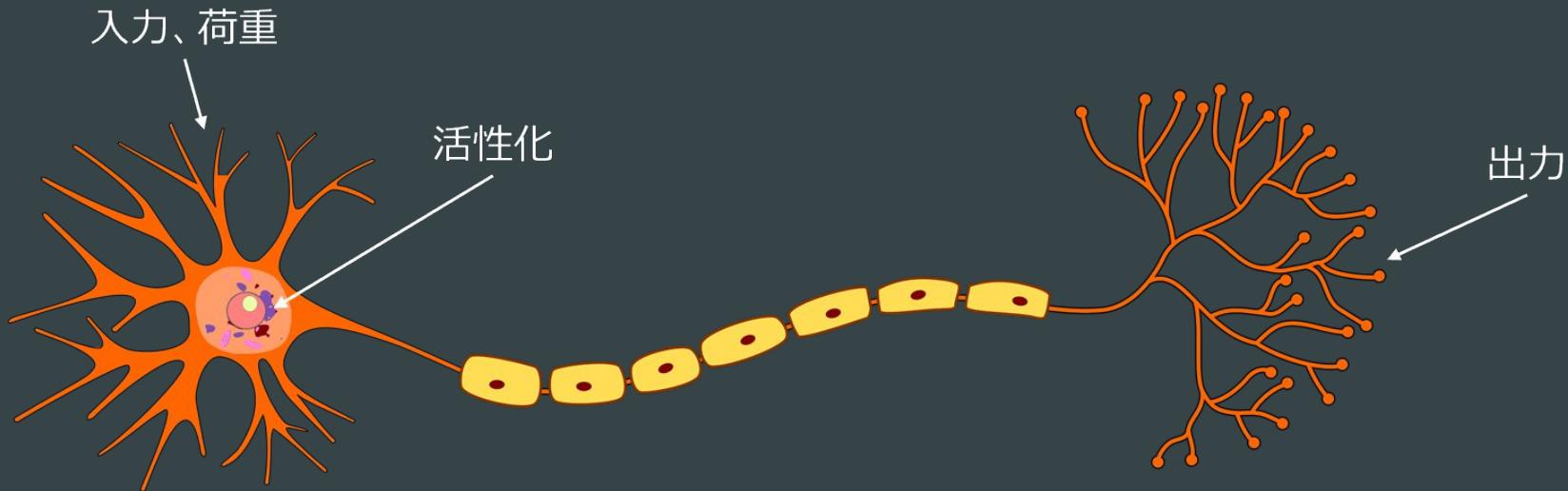
ニューラルネットワーク の仕組み



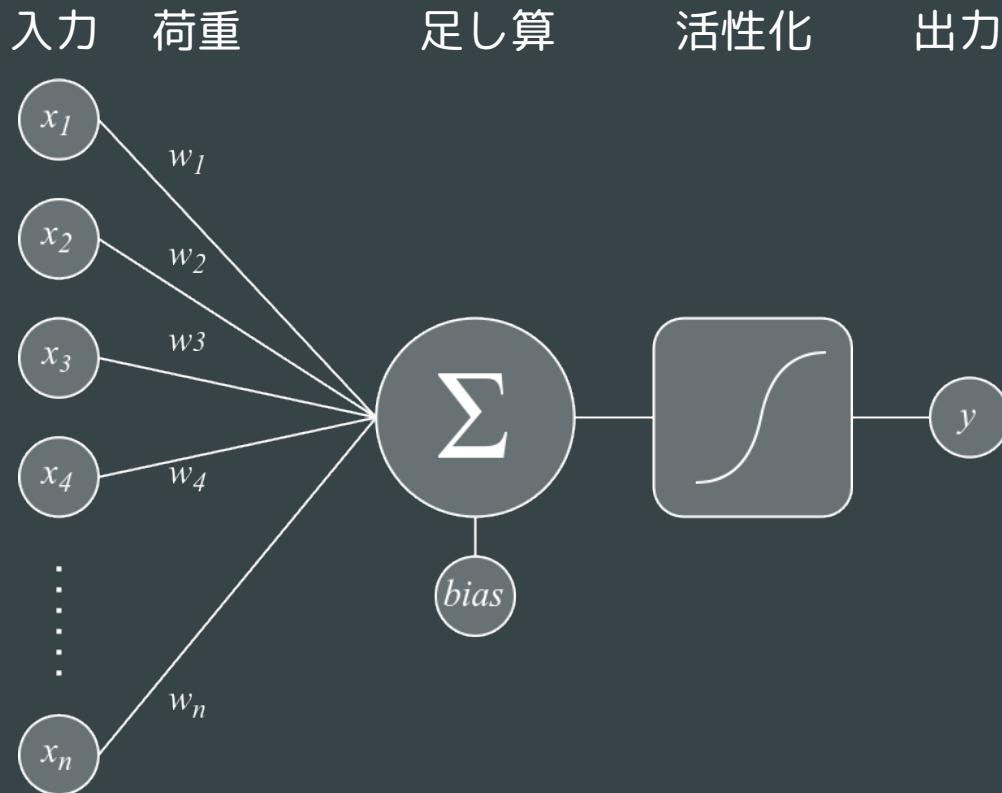
ニューラルネットワークの仕組み



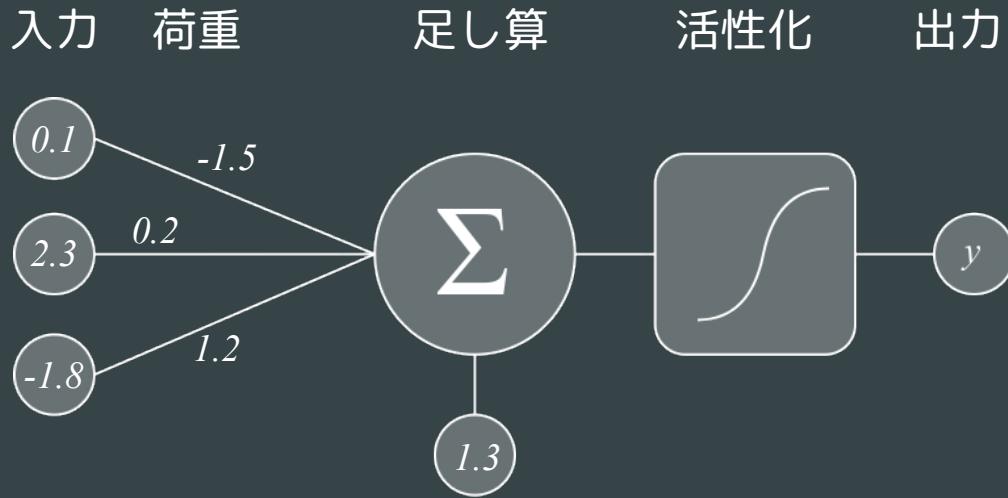
ニューラルネットワークの仕組み



ニューラルネットワークの仕組み



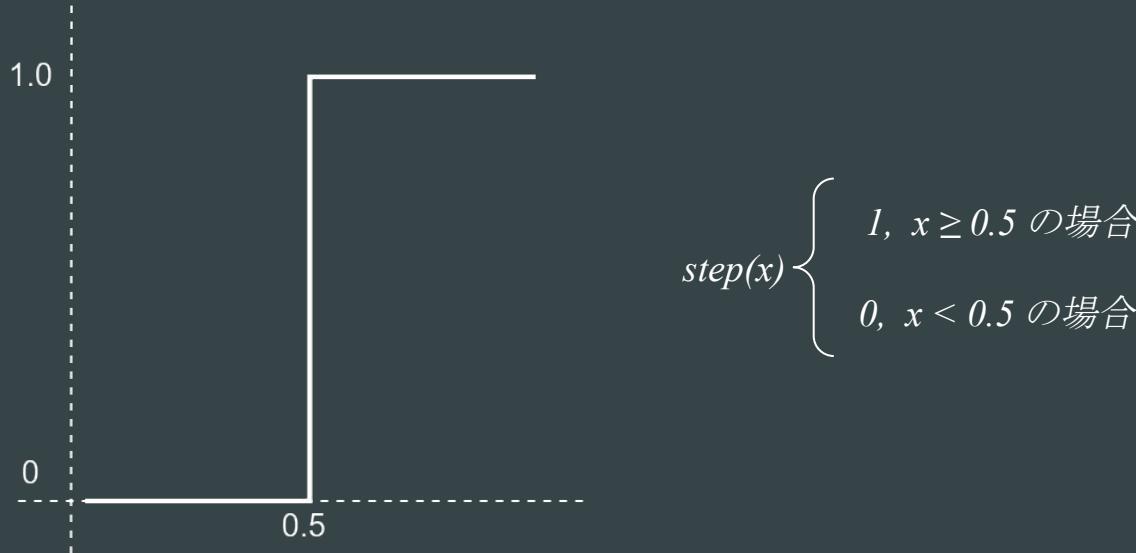
ニューラルネットワークの仕組み



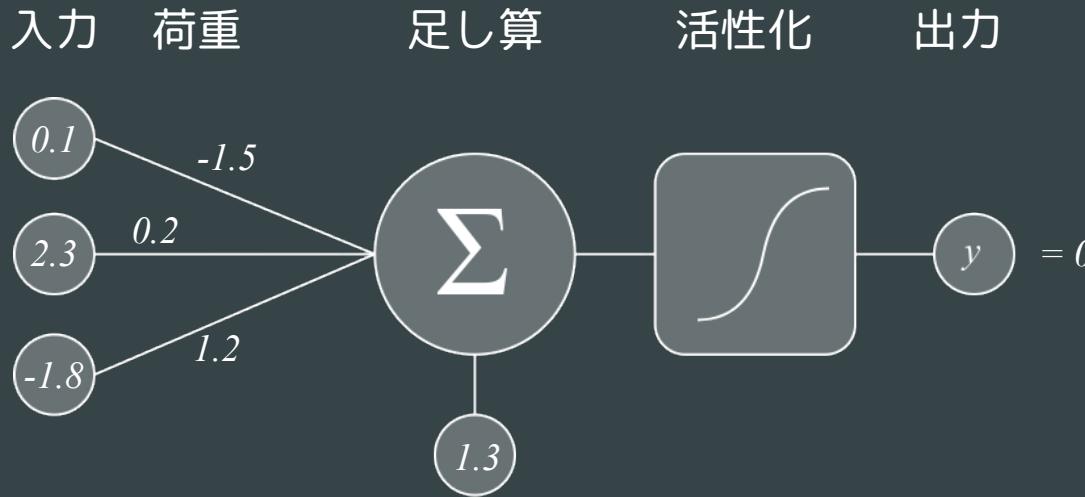
$$0.1 \times -1.5 + 2.3 \times 0.2 + -1.8 \times 1.2 + 1.3 = -0.55$$

ニューラルネットワークの仕組み

単純な活性化関数：ステップ関数

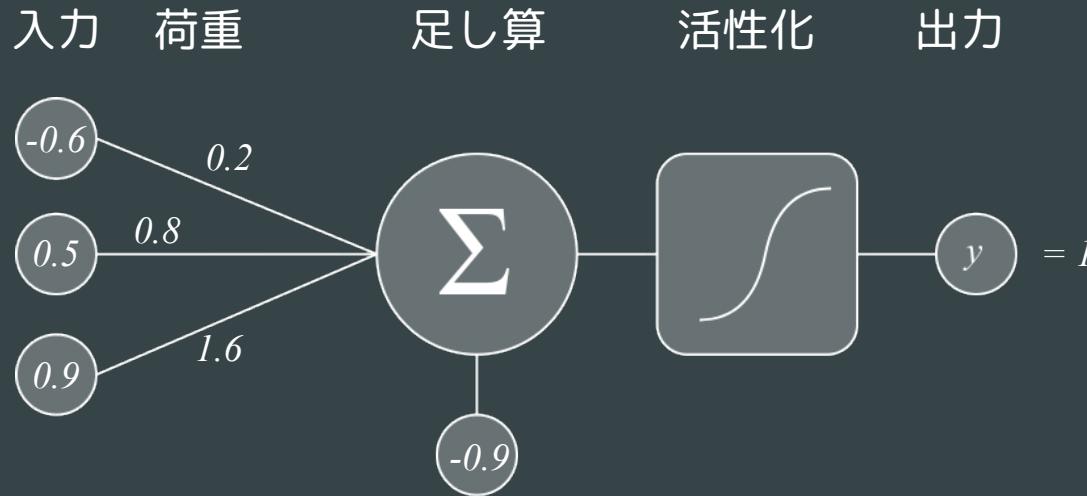


ニューラルネットワークの仕組み



$$0.1 \times -1.5 + 2.3 \times 0.2 + -1.8 \times 1.2 + 1.3 = -0.55 \rightarrow \text{step}(-0.55) = 0$$

ニューラルネットワークの仕組み



$$-0.6 \times 0.2 + 0.5 \times 0.8 + 0.9 \times 1.6 + -0.9 = 0.82 \rightarrow \text{step}(0.82) = 1$$

世界一最も単純な課題

で演習しよう！

ニューラルネットワークの仕組み

AND 演算子

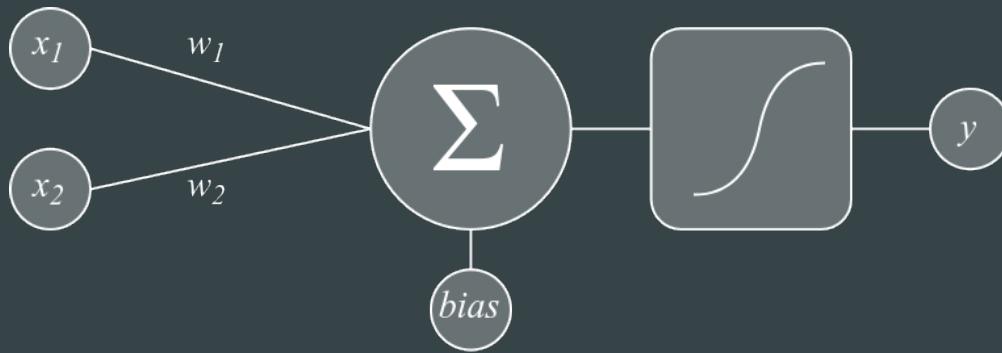
入力		出力
0	0	0
0	1	0
1	0	0
1	1	1

OR 演算子

入力		出力
0	0	0
0	1	1
1	0	1
1	1	1

ニューラルネットワークの仕組み

入力 荷重 足し算 活性化 出力



$$sum = x_1 \times w_1 + x_2 \times w_2 + bias$$

$$y = step(sum)$$

ニューラルネットワークの仕組み

- 言語：Python
- 具体的に：Anaconda Python
 - URL：<https://www.anaconda.com>
- その中で、Jupyter Notebookを使う
 - Python言語+Wikiを混ぜたツール
 - 開発しながらメモ、考えたことを記述できる
 - 勉強、説明、研究として便利
 - 本格的な開発ツールではない



世界で最も単純な課題：ANDとORを再現

それでは、演習へ～





ニューラルネットワーク の学習

学習の目的：

ネットワークの荷重を適切に更新する

でも、どの荷重が良いかわからない！

たしかに、最適の荷重がわからないが、
望ましい出力（ラベル）がわかる！

どれぐらい間違ったかを用い、
荷重を直せば良い！

少しづつ荷重を変え、
ニューラルネットワークの出力（推論）がよく
なるかどうかを確認すれば良いでしょう

…けど、どれぐらい変えれば良いか？

ランダムに変えても良いが、いつまでも終わらないので、もと賢い方法を使う

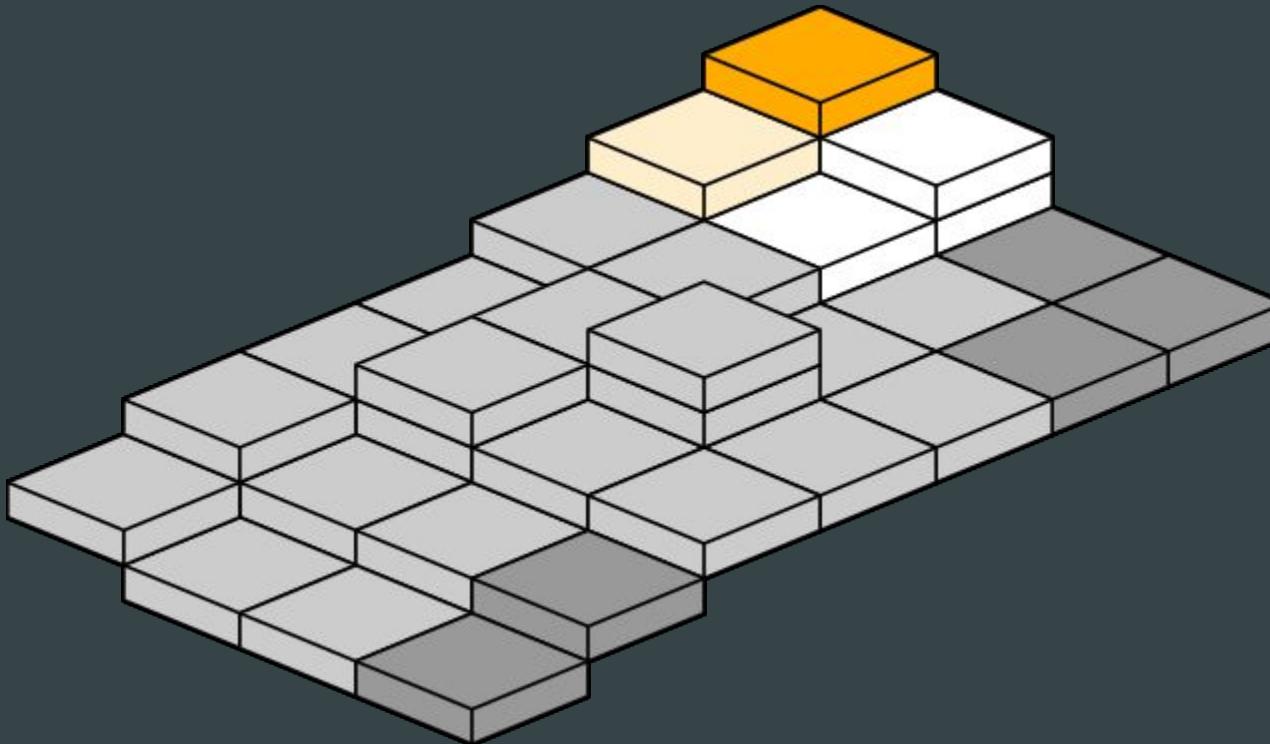
それは…

「勾配関数」 という

勾配関数の直感的なアイディア

濃い霧の山登り

ニューラルネットワークの学習



さて、この動きはPythonでどう実現する？

1. ランダムに初期
2. 推論
3. 出力とラベルの差分を計算
4. 誤差（段差）により、荷重を調整

$$w'_i = w_i + x_i \times error(y_{true}, y_{pred})$$

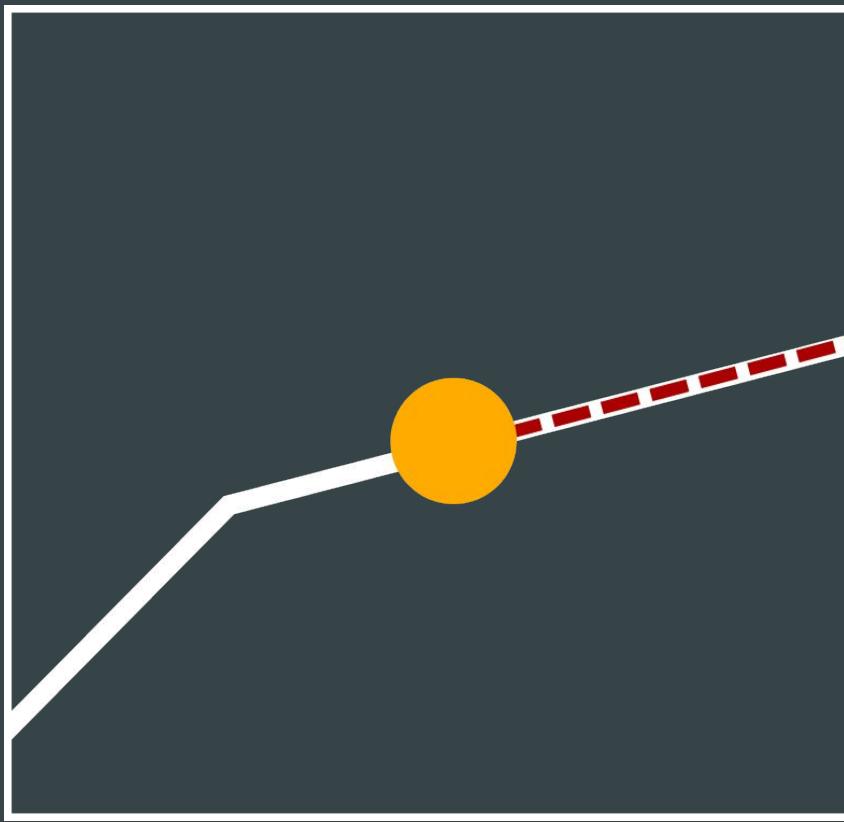
5. (2)に戻る

それでは、演習へ～

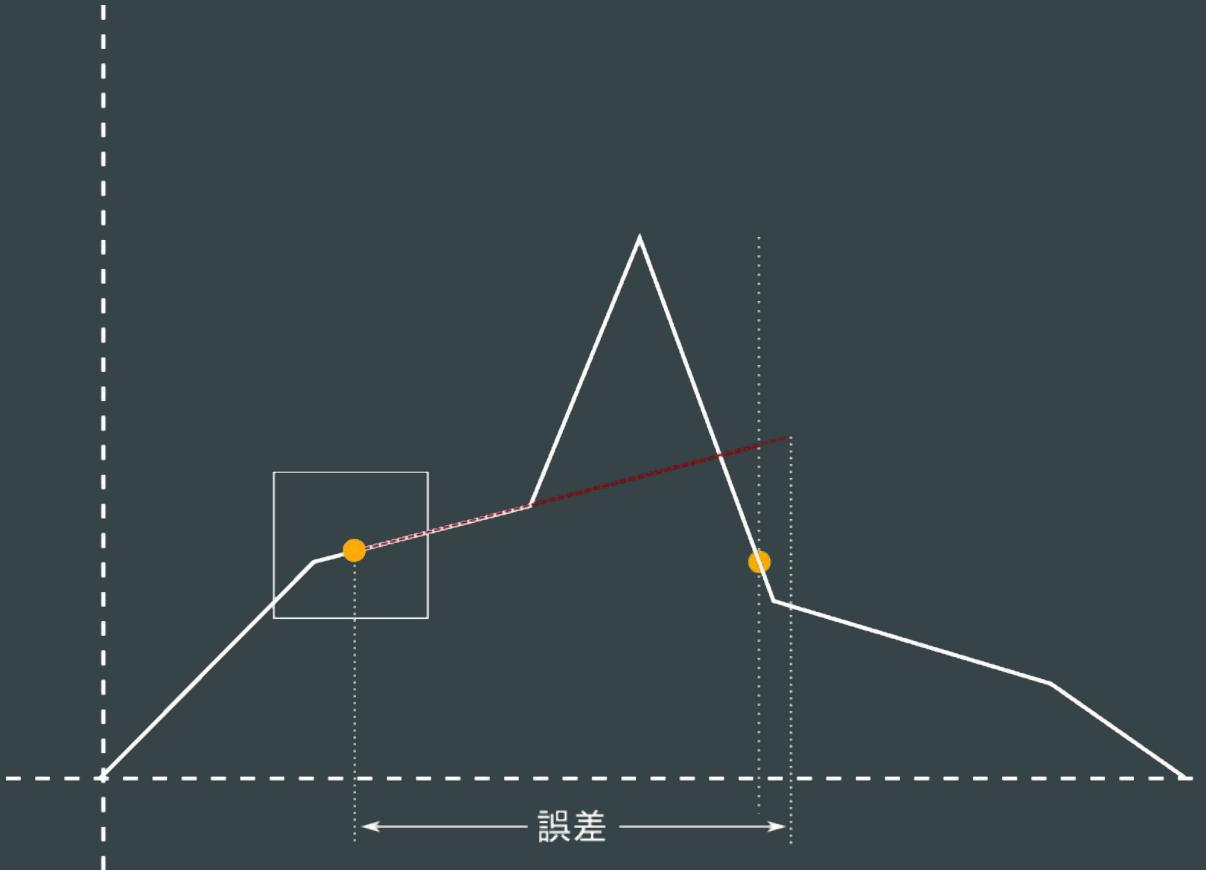


学習率とは

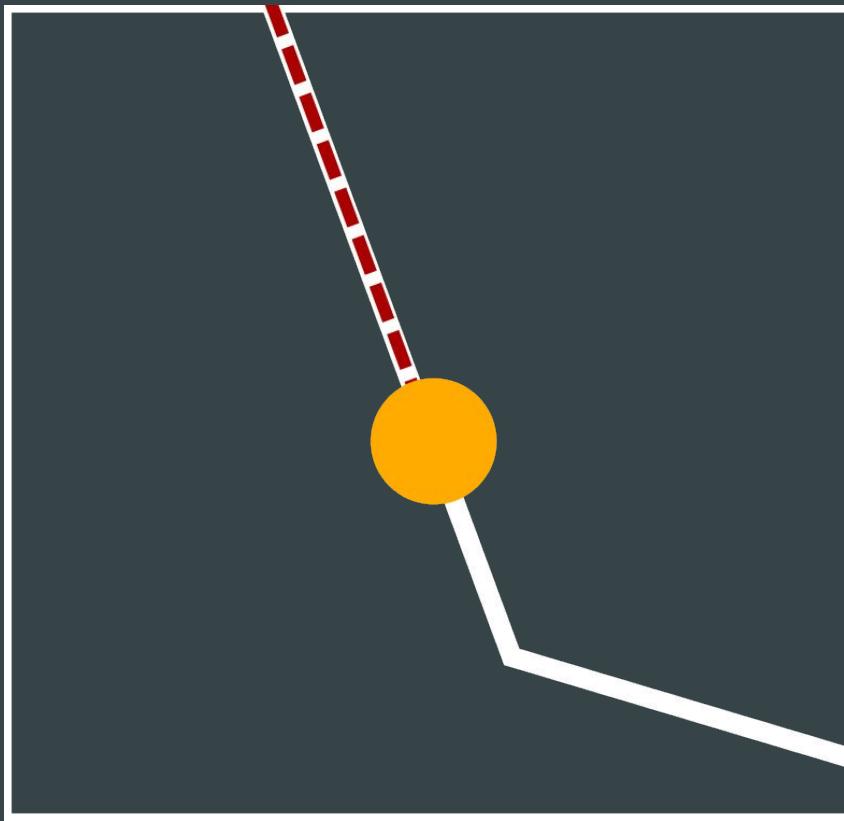
ニューラルネットワークの学習



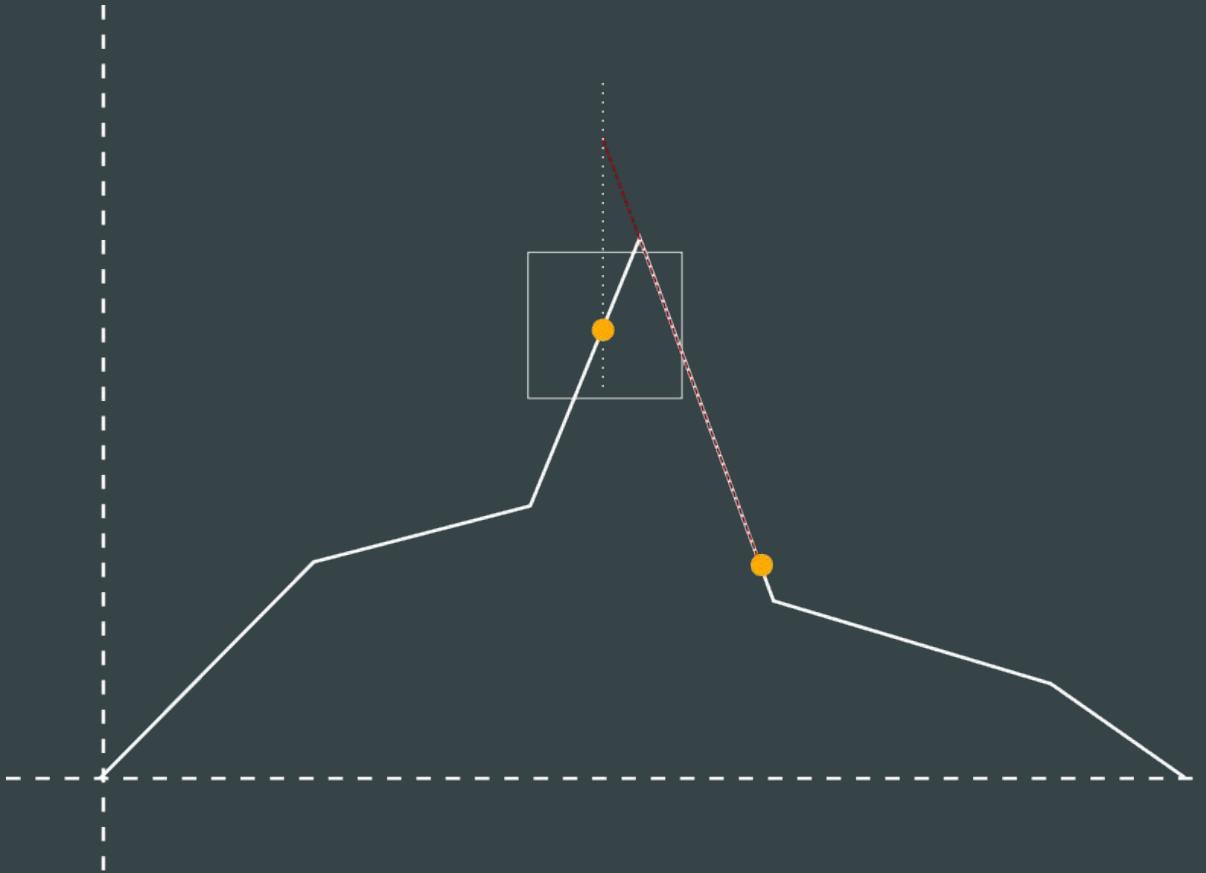
ニューラルネットワークの学習



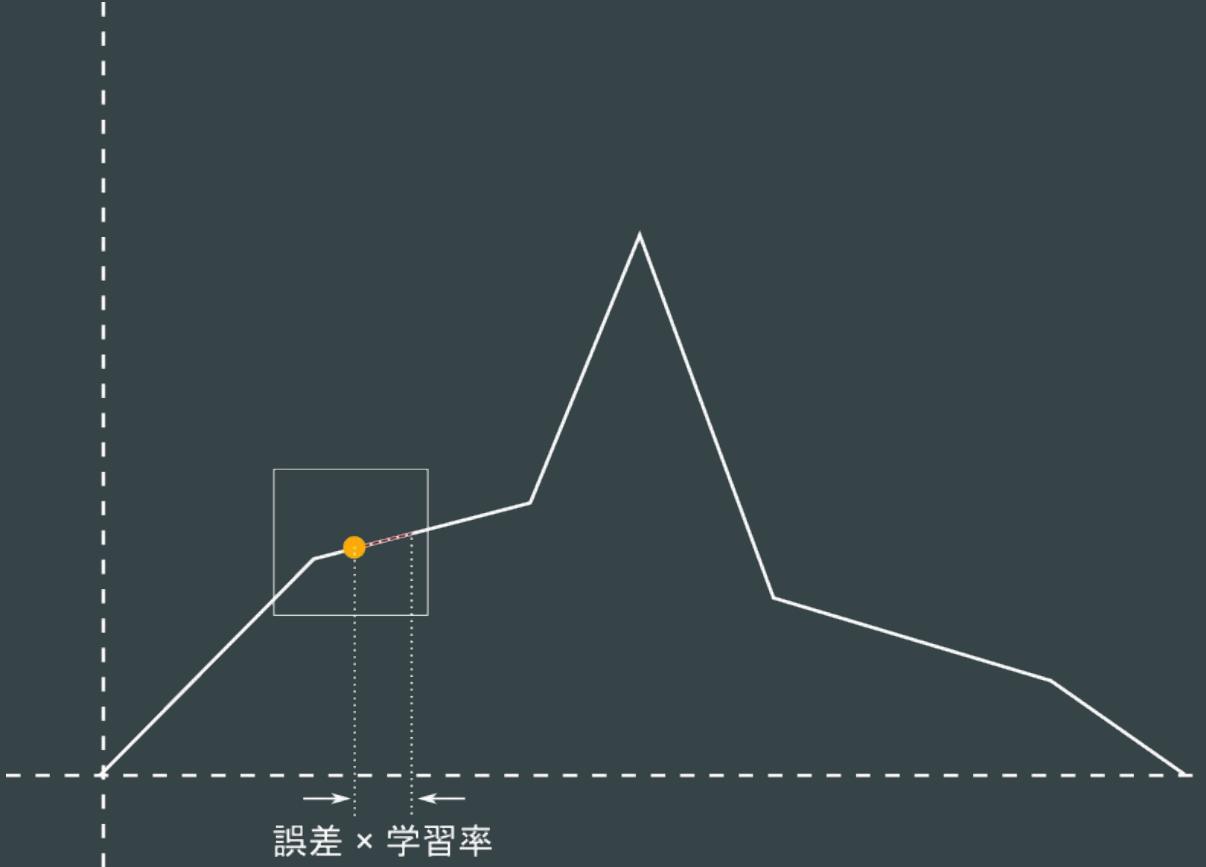
ニューラルネットワークの学習



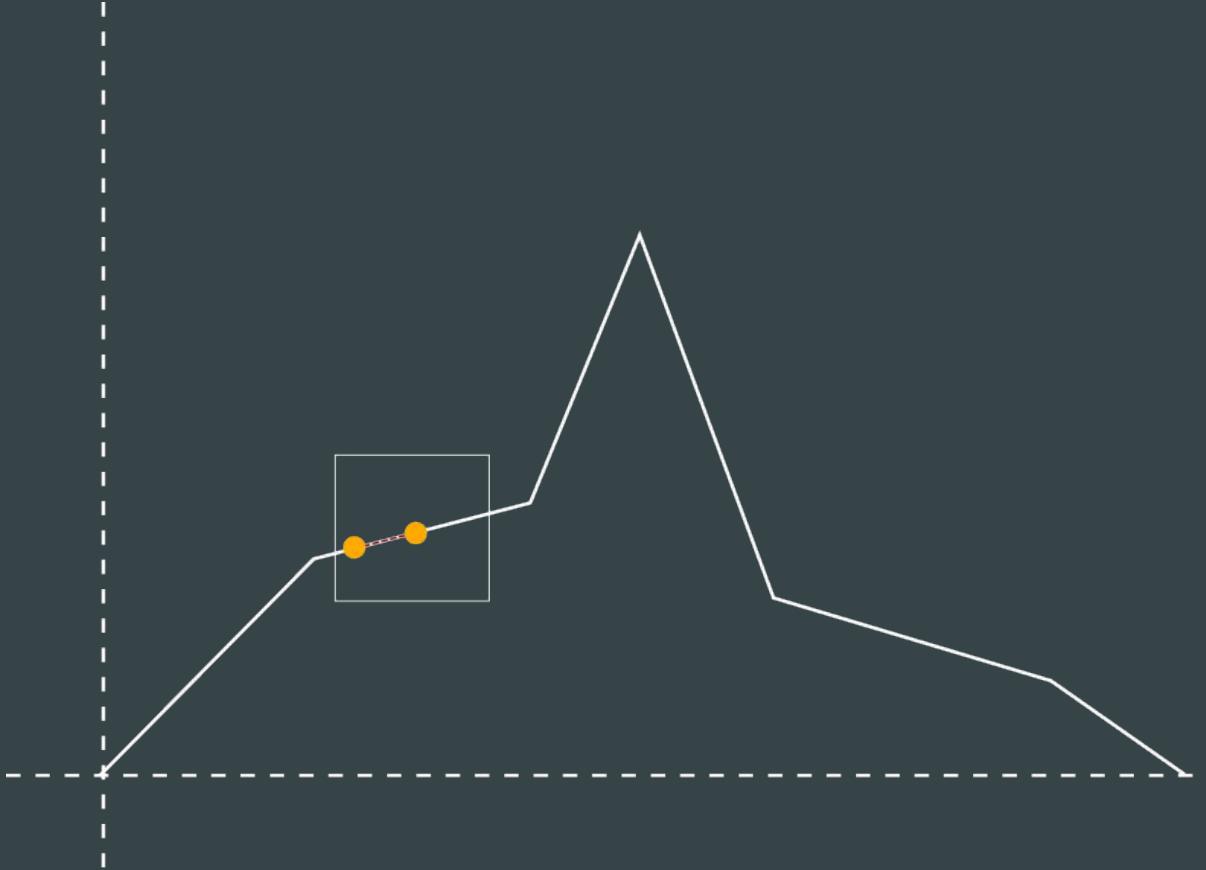
ニューラルネットワークの学習



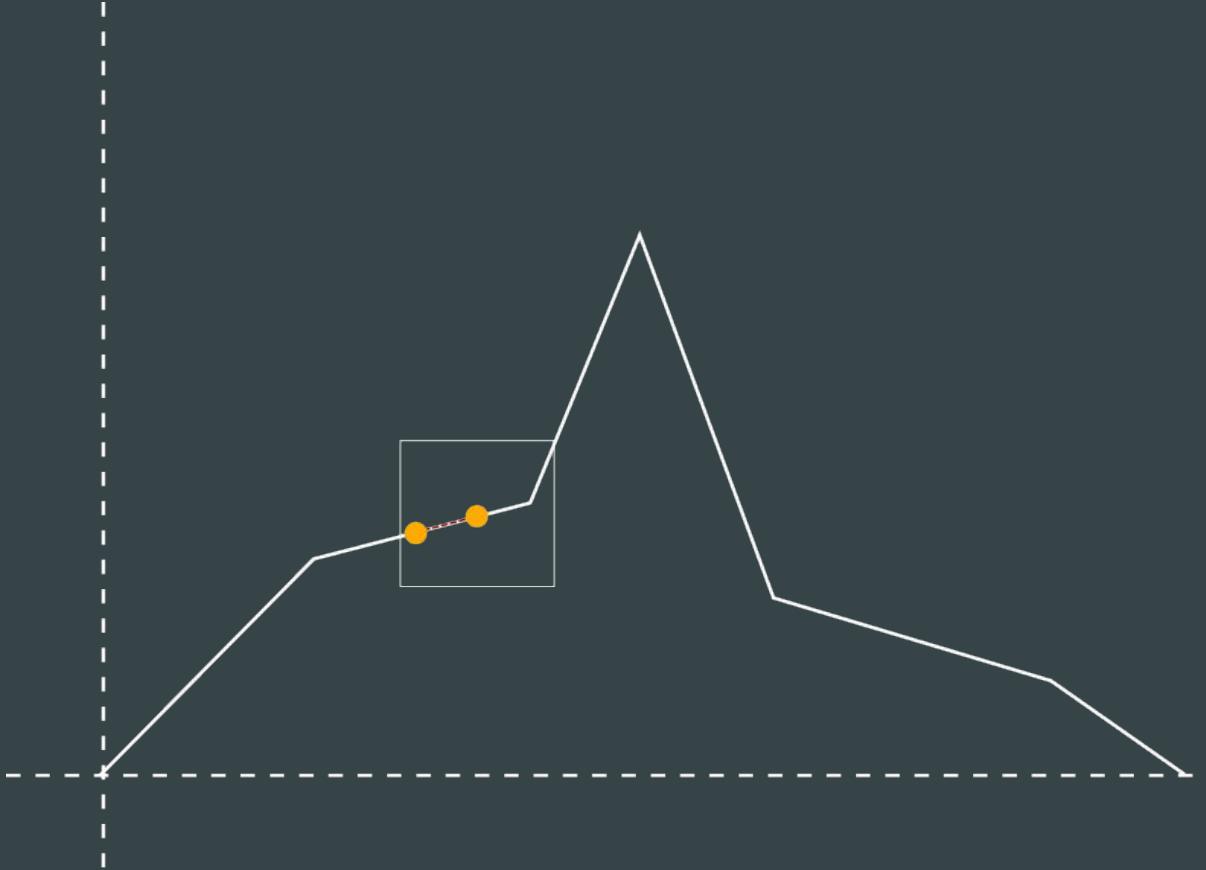
ニューラルネットワークの学習



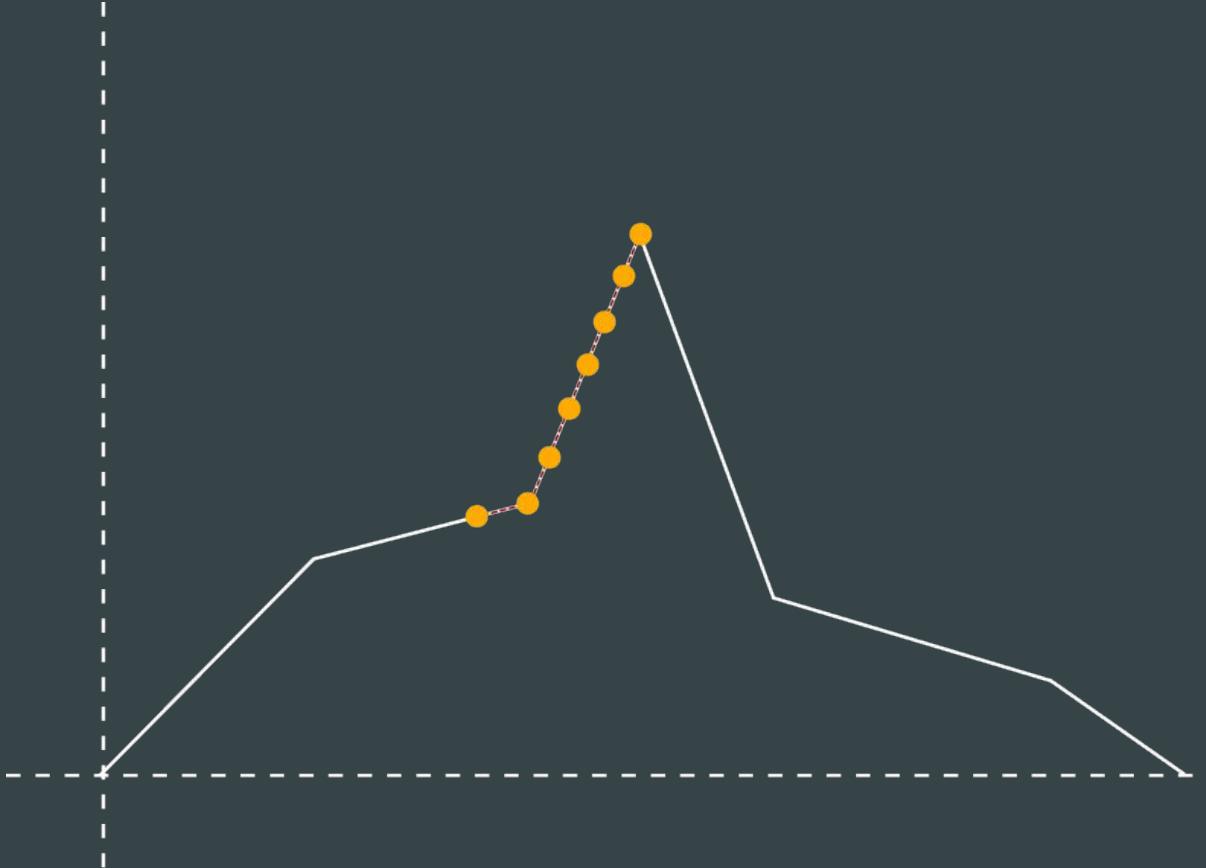
ニューラルネットワークの学習



ニューラルネットワークの学習



ニューラルネットワークの学習



K

Keras: ニューラルネットワークの開発 フレームワーク

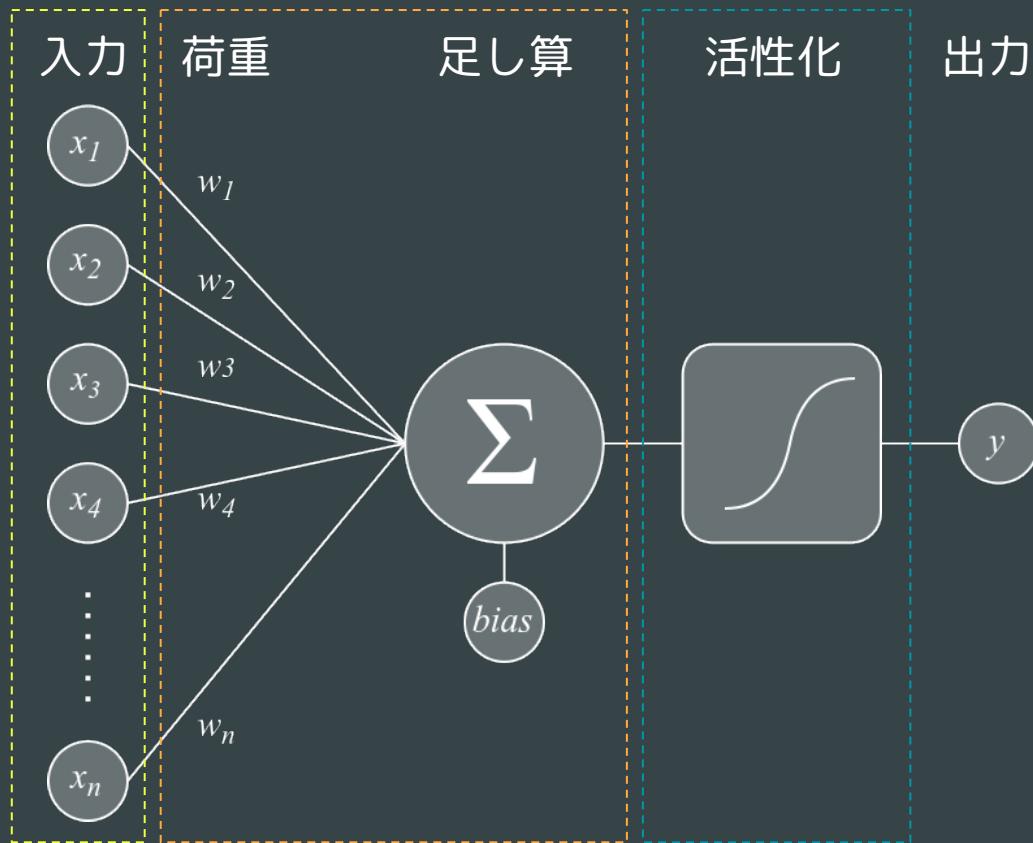
ニューラルネットワークに必要な「部品」簡単に組み合わせ、開発できるフレームワーク

Keras: ニューラルネットワークの開発フレームワーク

入力: Input

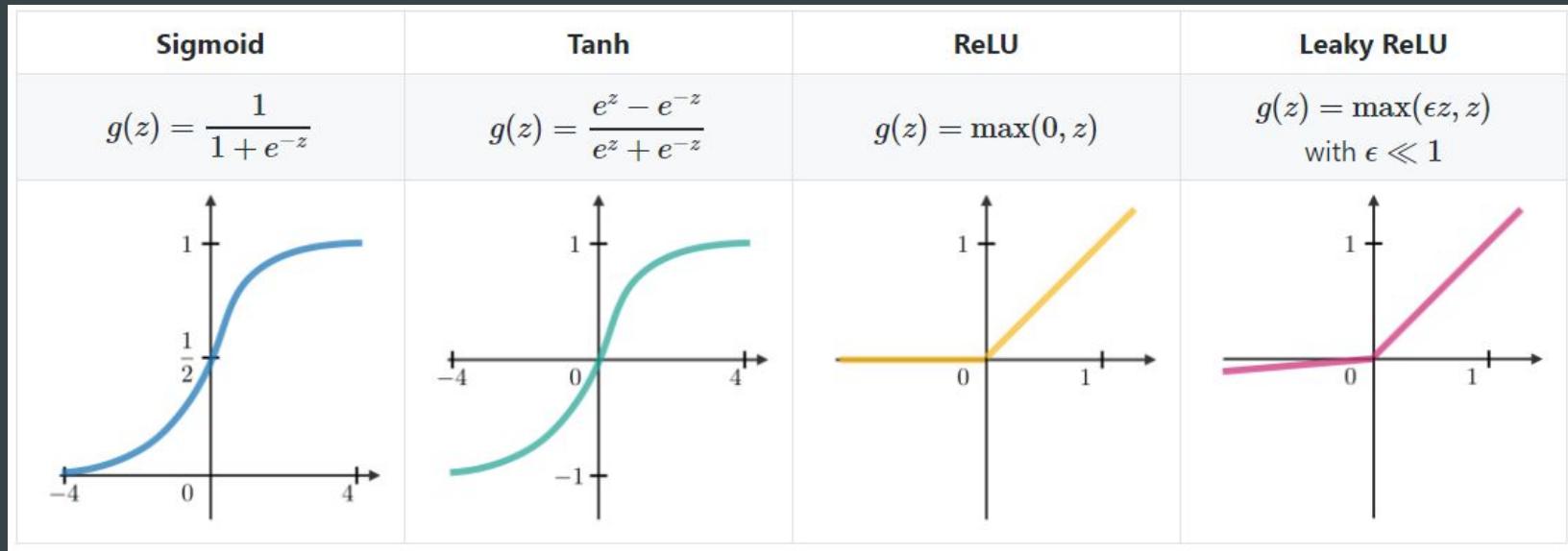
全結合レイヤ: Dense

活性化: Activation



ニューラルネットワークの仕組み

活性化関数



+
softmax (後で…)

注意：Step関数はない！

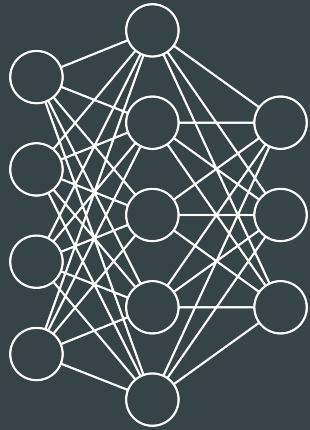
※ 勾配関数を計算するため、活性化関数の微分を求める必要があり、step関数の微分を求めることができないため、使うことができない。
代わりに「sigmoid」、または「tanh」をお使いください。

それでは、演習へ～



- ・ Kerasでネットワークを構築
- ・ ANDとORを解決
- ・ XORも

入力	出力
0	0
0	1
1	0
1	1



多層（ディープ） ネットワーク



なぜXORは失敗するのか？

ネットワークの計算式を解析してみると：

$$x_1w_1 + x_2w_2 + bias$$



XORは失敗する原意：
非線形の問題であるため

では、直線で分類できない
問題をどう解決する？

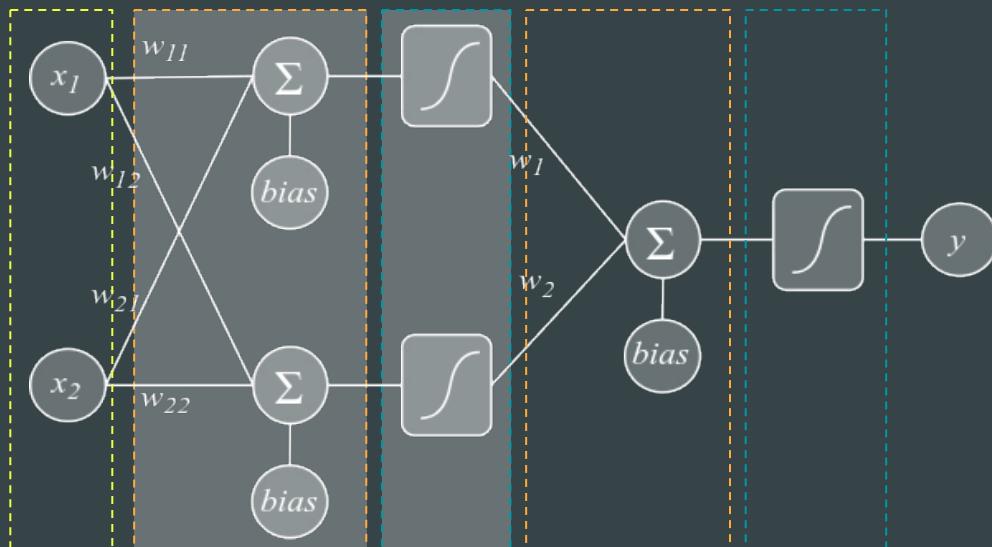
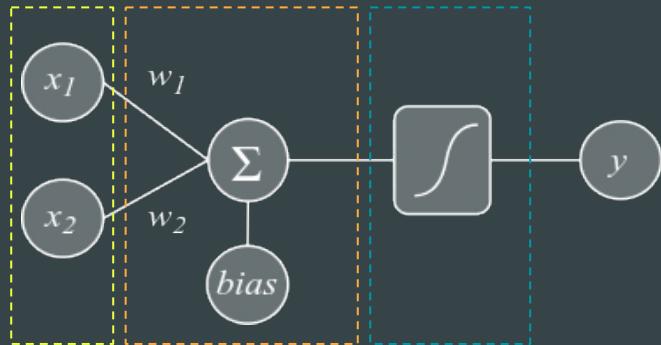
答え：ネットワークも非線形にする

=

中間レイヤを挿入

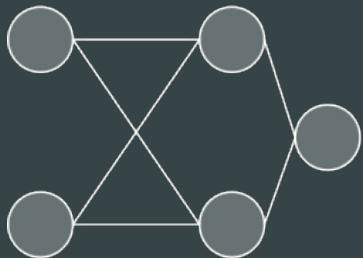
多層（ディープ）ネットワーク

Input
Dense
Activation
中間層

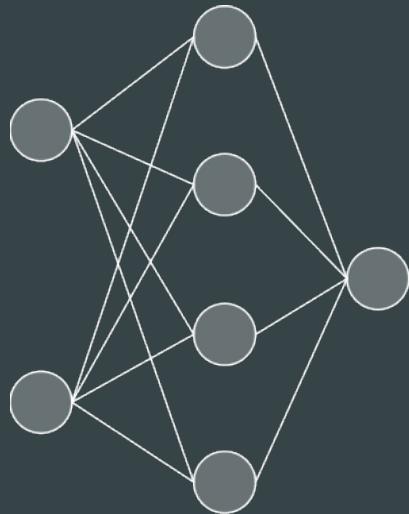


多層（ディープ）ネットワーク

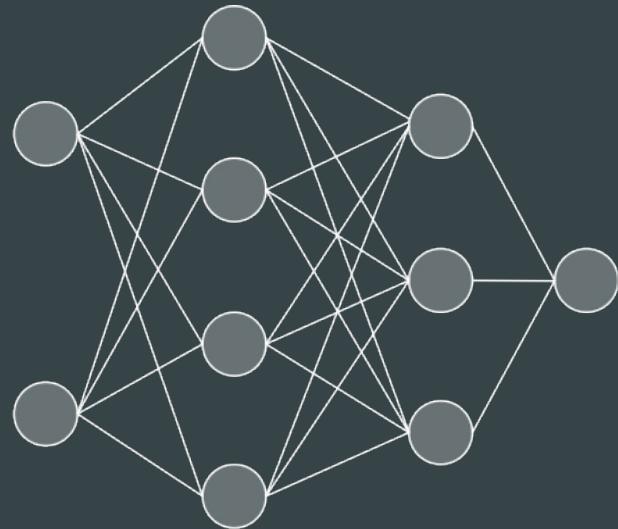
様々な構成ができる！



入力:2個
中間数:2個
出力:1個



入力:2個
中間数:4個
出力:1個



入力:2個
中間数1:4個
中間数2:3個
出力:1個

多層（ディープ）ネットワーク

それでは、演習へ～





前處理 · 後處理



今まで、の入力と出力は「0」 or 「1」
けど、実データは中々そうではない

- 例：パスワードを分類する
- 入力：パスワード
 - 英数字、記号
- 出力：強さ
 - 弱い・やや弱い・普通・やや強い・強い

AIエンジニアの一つの仕事：
データを解析、前処理し、
ネットワークの入力に合わせる

データの「特徴」はなに？

パスワードの「特徴」

長さ : 0~20字

小文字 : 0~20字

大文字 : 0~20字

数字 : 0~20字

記号があるかないか : 0 or 1

正規化とは

- 各入力の範囲をコントロール
 - 例：人間データ
 - 入力A：体重 - 範囲：2kg ~ 200kg
 - 入力B：身長 - 範囲：0.3m ~ 2.5m
- 学習しやすくなる
 - 荷重の調整は小さなステップができる

入力の前処理

- 長さの最大値を20にする
 - 長さを20で割る
- 記号はフラグで0 or 1にする

1qA2wsEed

[0.45 0.25 0.1 0.1 0]

長さ：9文字
正規化： $9/20 = 0.45$

小文字：5文字
正規化： $5/20 = 0.25$

大文字：2文字
正規化： $2/20 = 0.1$

数字：2文字
正規化： $2/20 = 0.1$

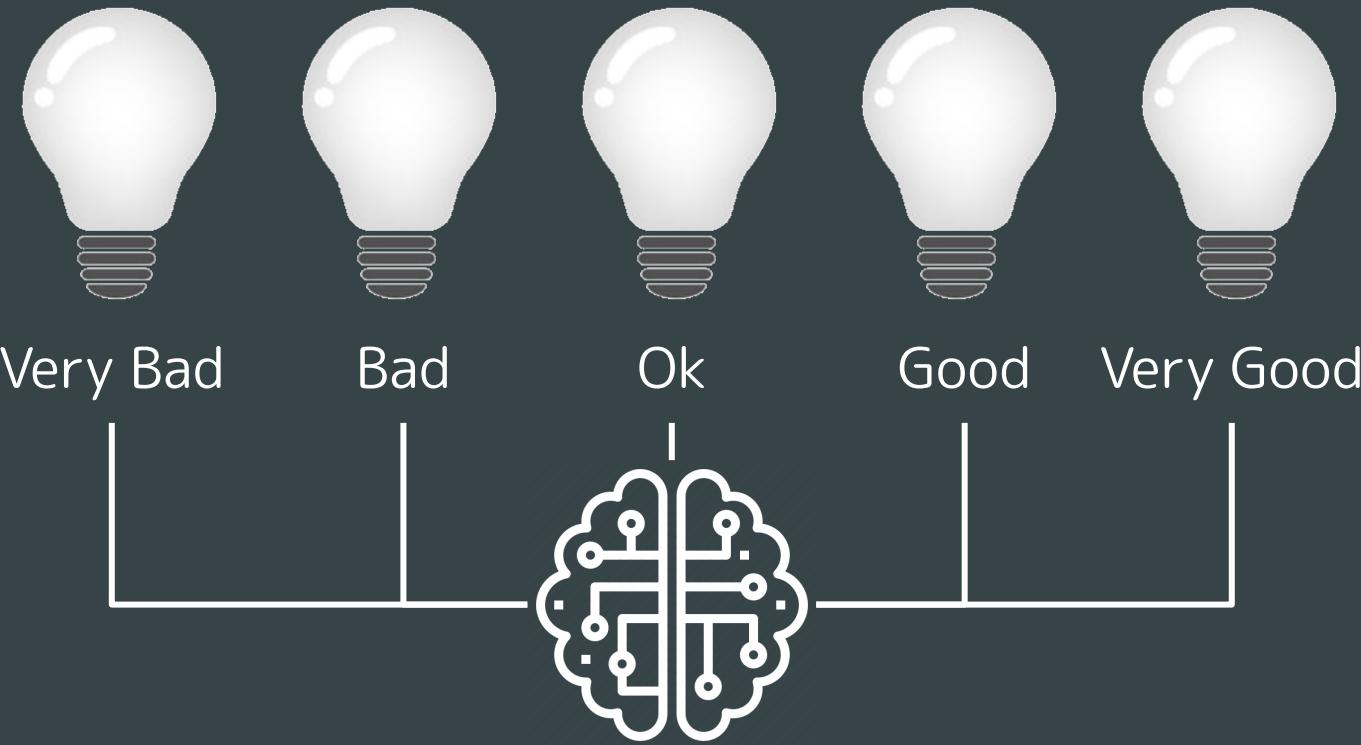
記号：なし (0)

出力の前処理

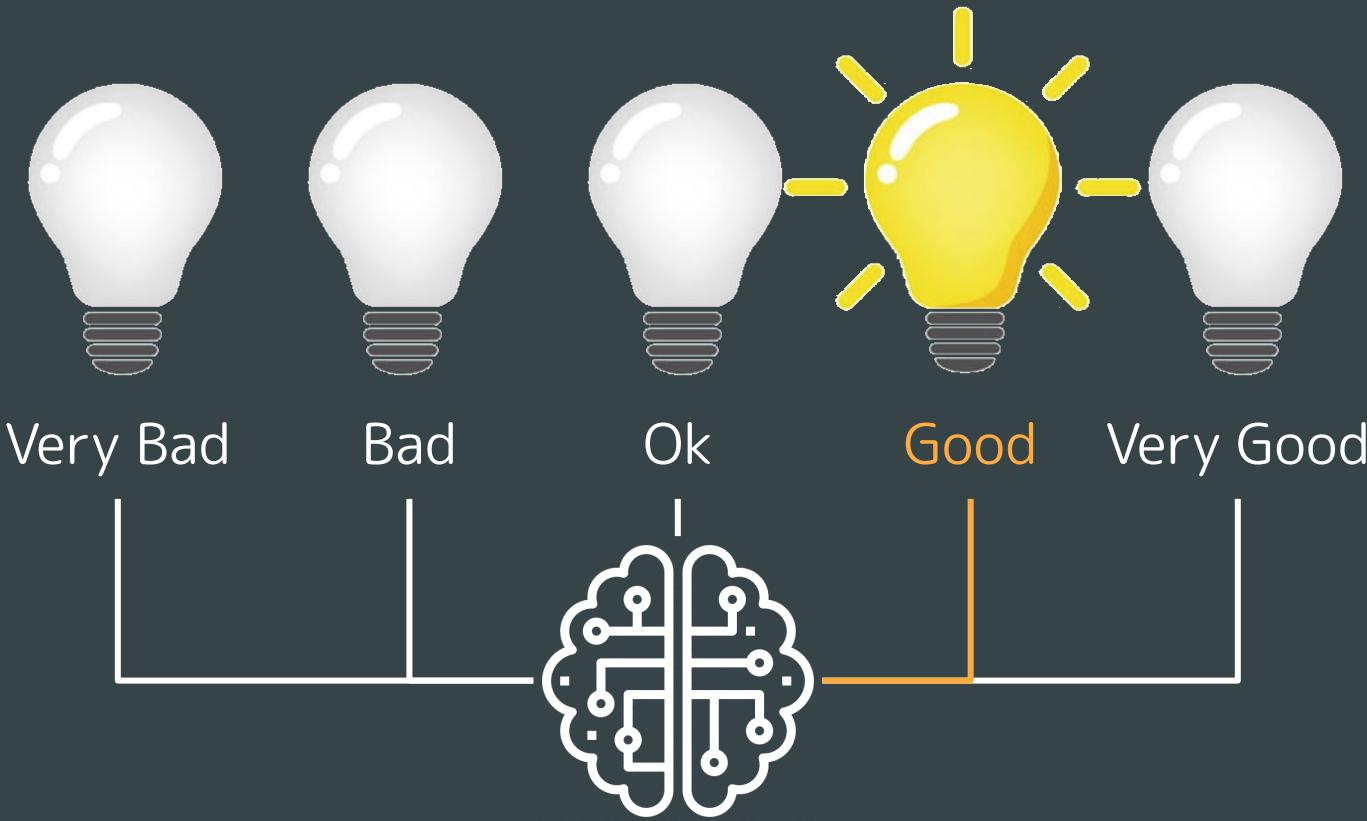
- 0: Very Bad
- 1: Bad
- 2: Ok
- 3: Good
- 4: Very Good

- 「分類課題」 → One Hot Encoding

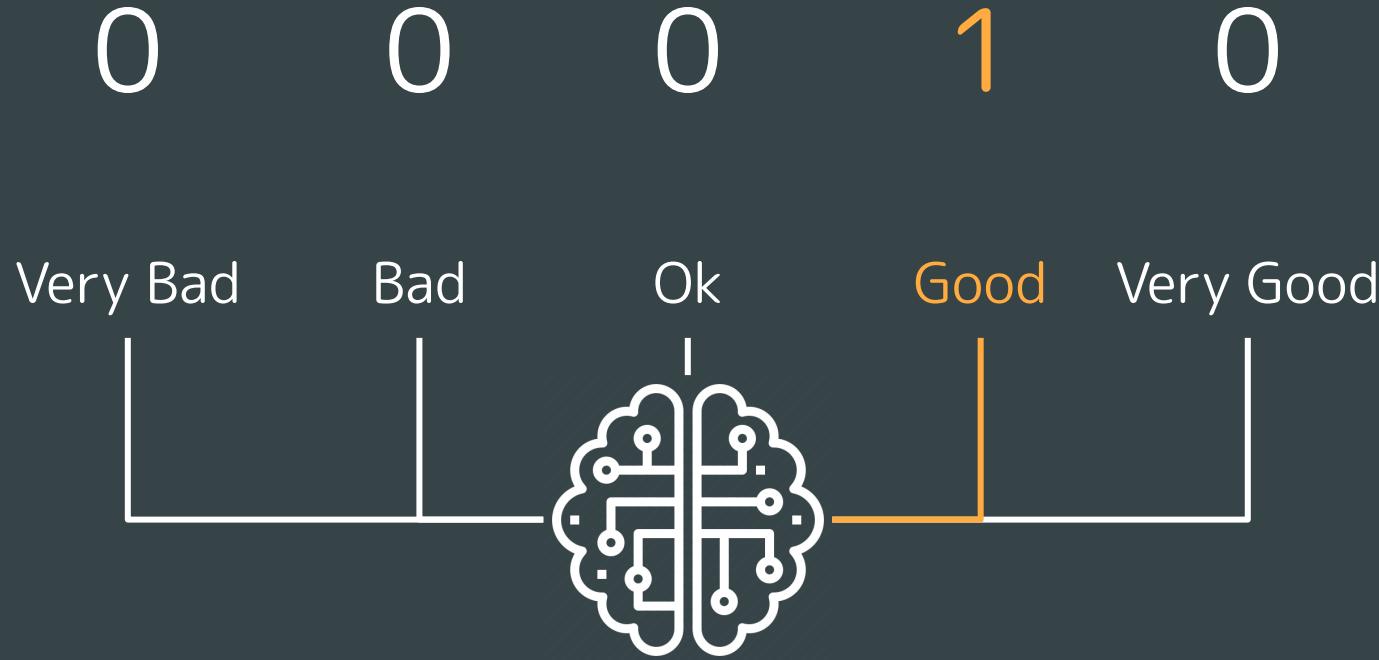
前處理 · 後處理



前處理 · 後處理



前處理 · 後處理

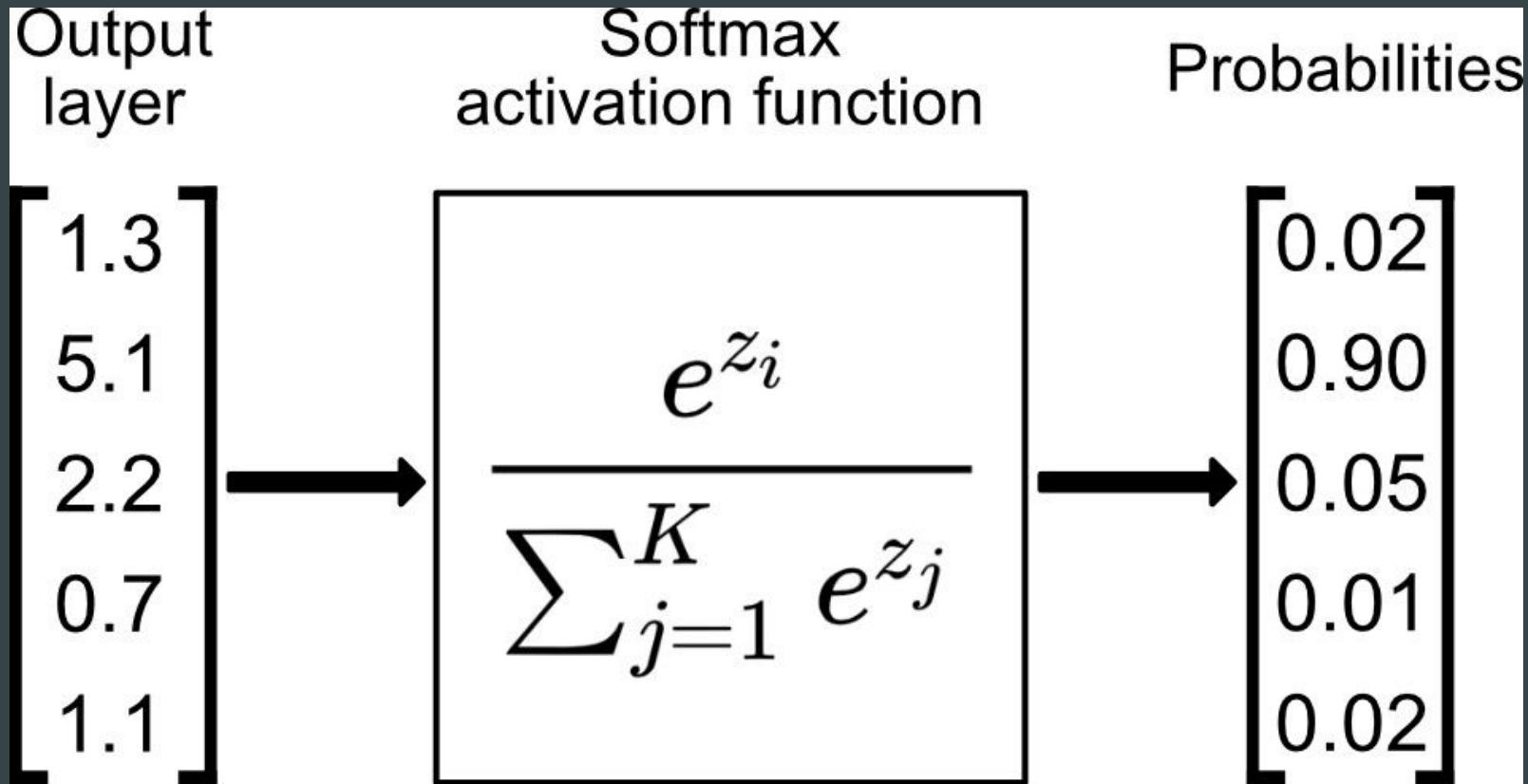


前処理・後処理

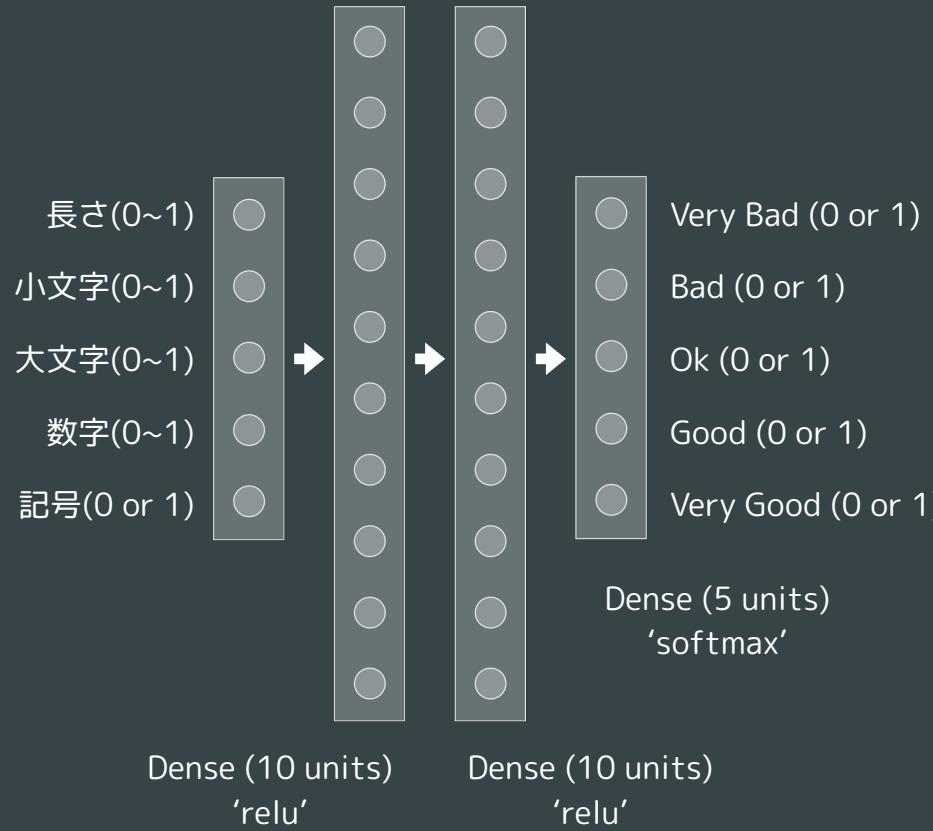
ラベル	数値化	One-Hot Encoding
Very Bad	0	0 0 0 0 1
Bad	1	0 0 0 1 0
Ok	2	0 0 1 0 0
Good	3	0 1 0 0 0
Very Good	4	1 0 0 0 0

残っていた活性化関数
softmax

レイヤ（多数ニューロン）の出力
を「確率」に変換



前処理・後処理



後処理は？

それでは、演習へ～

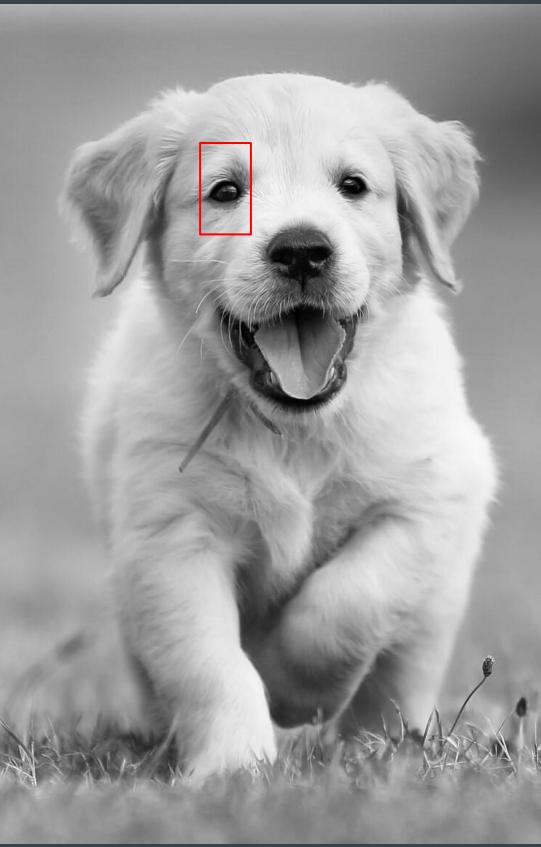




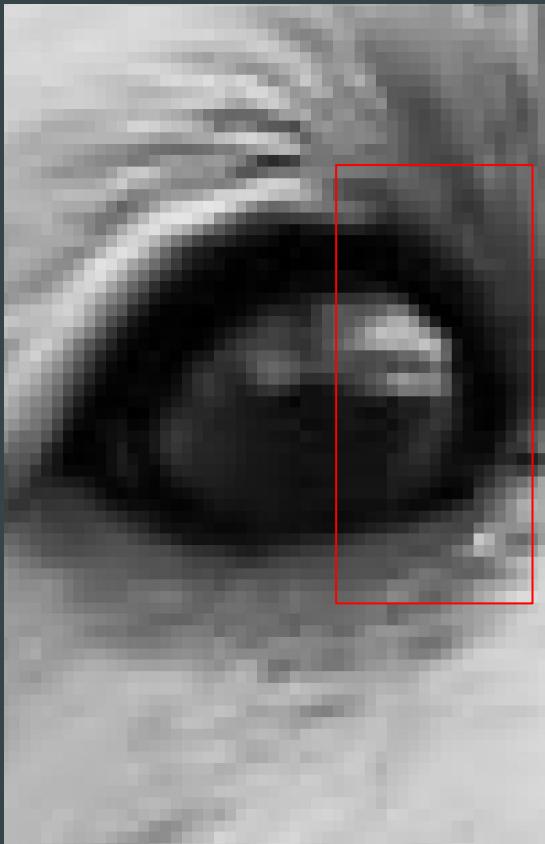
画像認識①

AIは画像でも学習できる！
ただ、適切に前処理が必要

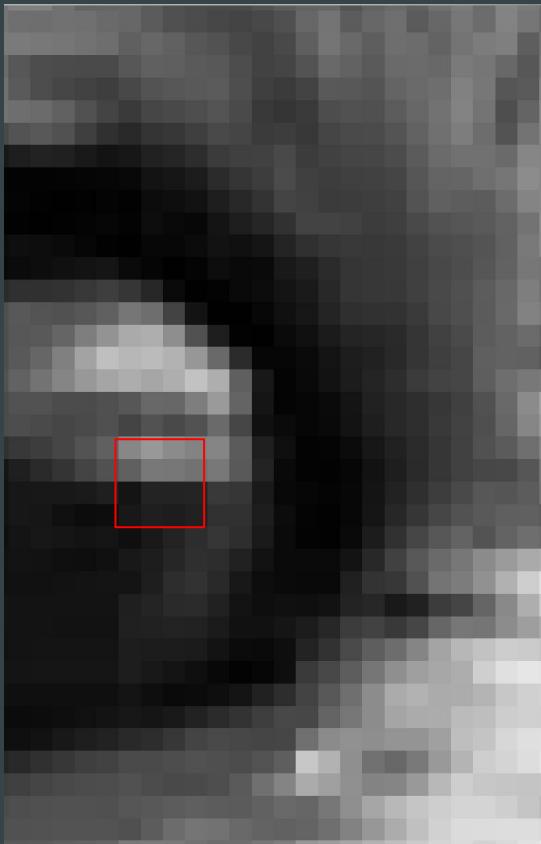
画像認識①



画像認識①

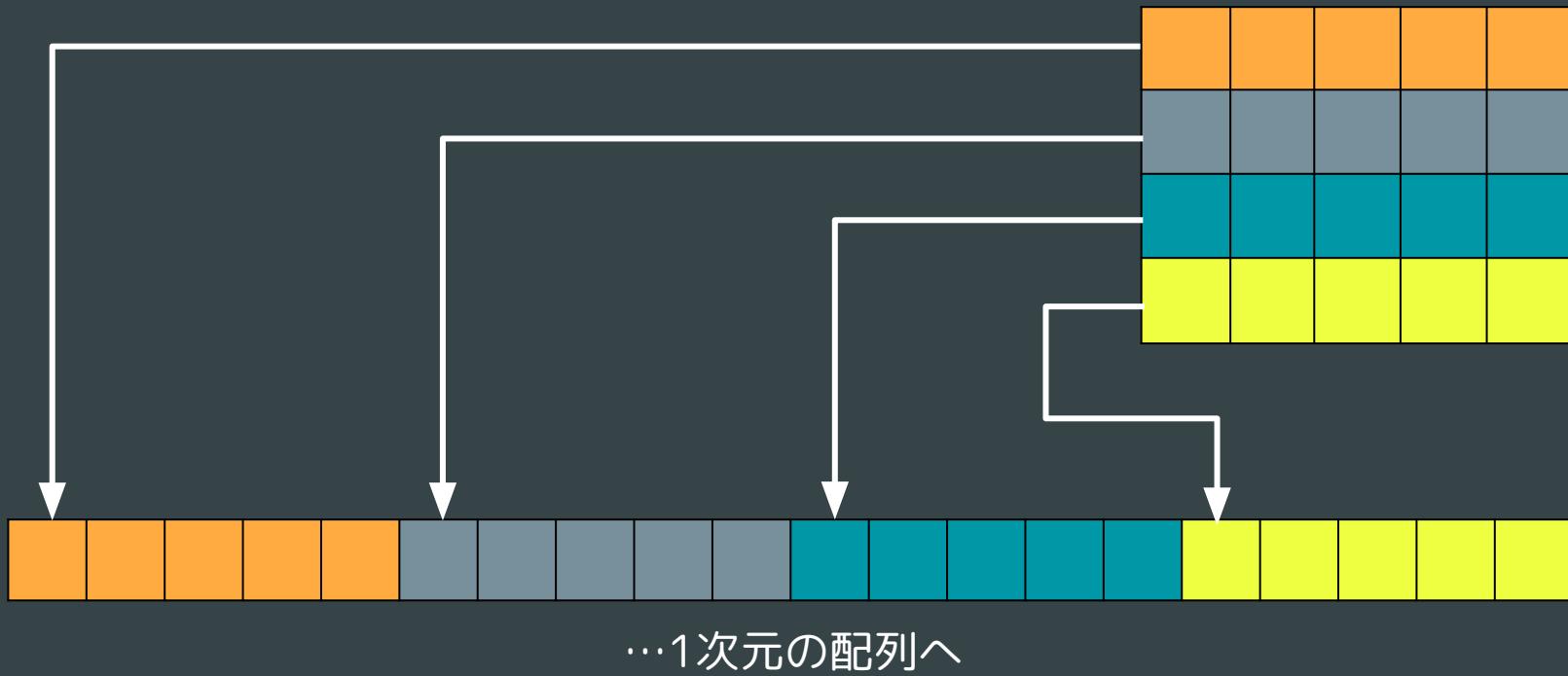


画像認識①



135	152	139	131
96	120	119	114
20	35	35	36
24	31	29	33

2次元の行列を…



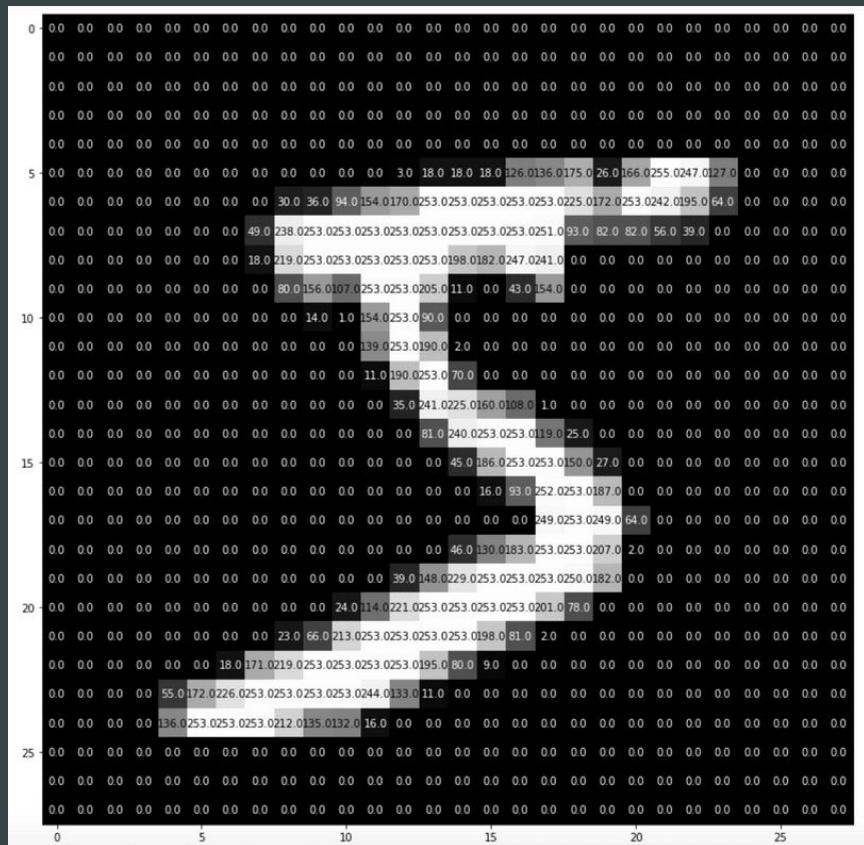
画像認識①

今回の課題
手書きの数字を認識

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

データセットの紹介：

- MNIST(エムニスト) という
- 0~9の数字
- 画像は70,000枚
 - 学習用：60,000枚
 - 検証用：10,000枚
- サイズが28×28ピクセルの画像
 - 1次元にすると：784
- グレースケール
 - 各ピクセルの明度は0~255で表現

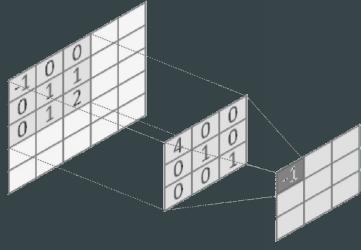


やらないといけないこと：

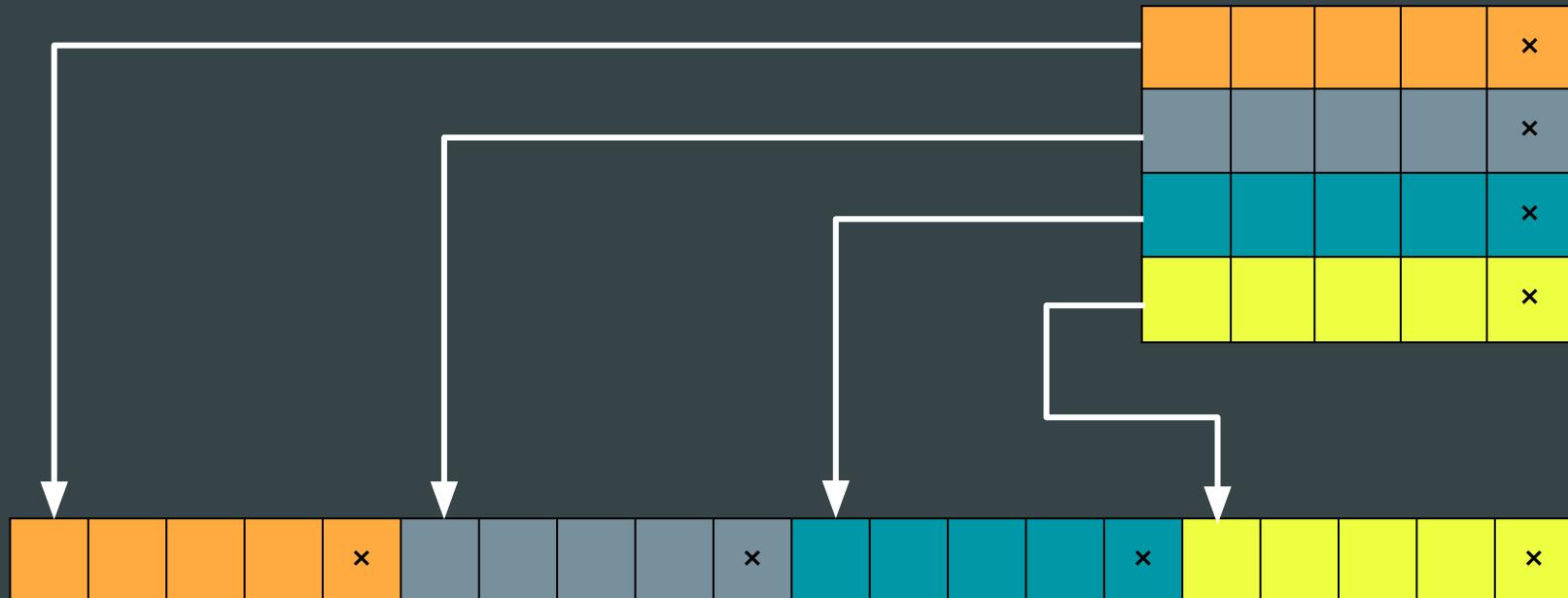
- 2次元の画像を1次元に変換
- 正規化 ($0\sim255 \rightarrow 0\sim1$)
- 「0~9」のラベルをOne-Hotに変換
- ニューラルネットワークの構築
- 学習
- 精度検証

それでは、演習へ～





ここに大きな弱点がある



隣なのに、離れている

相対的な関係を保持できると良い…

新しいレイヤの紹介： Convolution（畳み込み）

入力

x_1	x_2	x_3	x_4	...		

フィルタ

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

画像認識②

入力

20	5	4	25	10		
8	4	14	21	3	...	
5	12	3	35	40		
		...				

出力

	23					

フィルタ

0	1	0
0.5	1	0.5
0	0	1

$$\begin{aligned} & 0 + 5 + 0 + 4 + 8 \times 0.5 + 4 \times 1 + 14 \times 0.5 + 5 \times 0 + 12 \times 0 + 3 \times 1 \\ & 20 \times 0 + 5 \times 1 + 4 \times 0 + 8 \times 0.5 + 4 \times 1 + 14 \times 0.5 + 5 \times 0 + 12 \times 0 + 3 \times 1 \end{aligned}$$

画像認識②

入力

20	5	4	25	10		
8	4	14	21	3	...	
5	12	3	35	40		
		...				

出力

	23	65				

フィルタ

0	1	0
0.5	1	0.5
0	0	1

$$\begin{aligned} & 0 + 4 + 0 + 2 + 14 + 10.5 + 0 + 0 + 0 + 35 \\ & 5 \times 0 + 4 \times 1 + 25 \times 0 + 4 \times 0.5 + 14 \times 1 + 21 \times 0.5 + 12 \times 0 + 3 \times 0 + 35 \times 1 \end{aligned}$$

画像認識②

入力

20	5	4	25	10		
8	4	14	21	3	...	
5	12	3	35	40		
		...				

出力

23	65	94				

フィルタ

0	1	0
0.5	1	0.5
0	0	1

$$\begin{aligned} & 0 + 25 + 0 + 7 + 21 + 1.5 + 0 + 0 + 0 + 40 \\ & 4 \times 0 + 25 \times 1 + 10 \times 0 + 14 \times 0.5 + 21 \times 1 + 3 \times 0.5 + 3 \times 0 + 35 \times 0 + 40 \times 1 \end{aligned}$$

画像認識②

输入

20	5	4	25	10	
8	4	14	21	3	...
5	12	3	35	40	
		...			

出力

23	65	94		

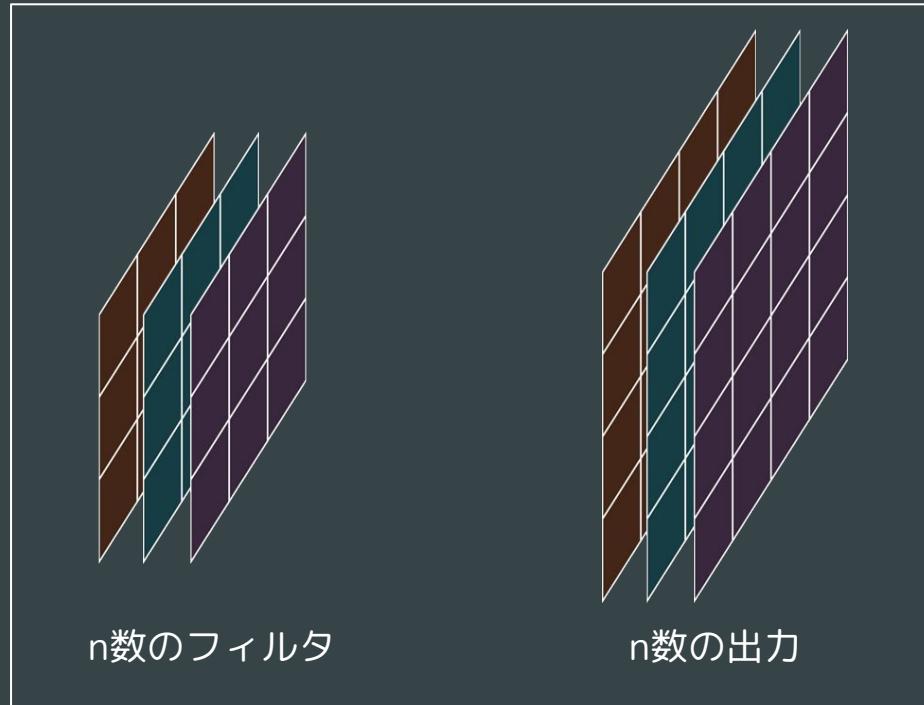
フィルタ

0	1	0
0.5	1	0.5
0	0	1

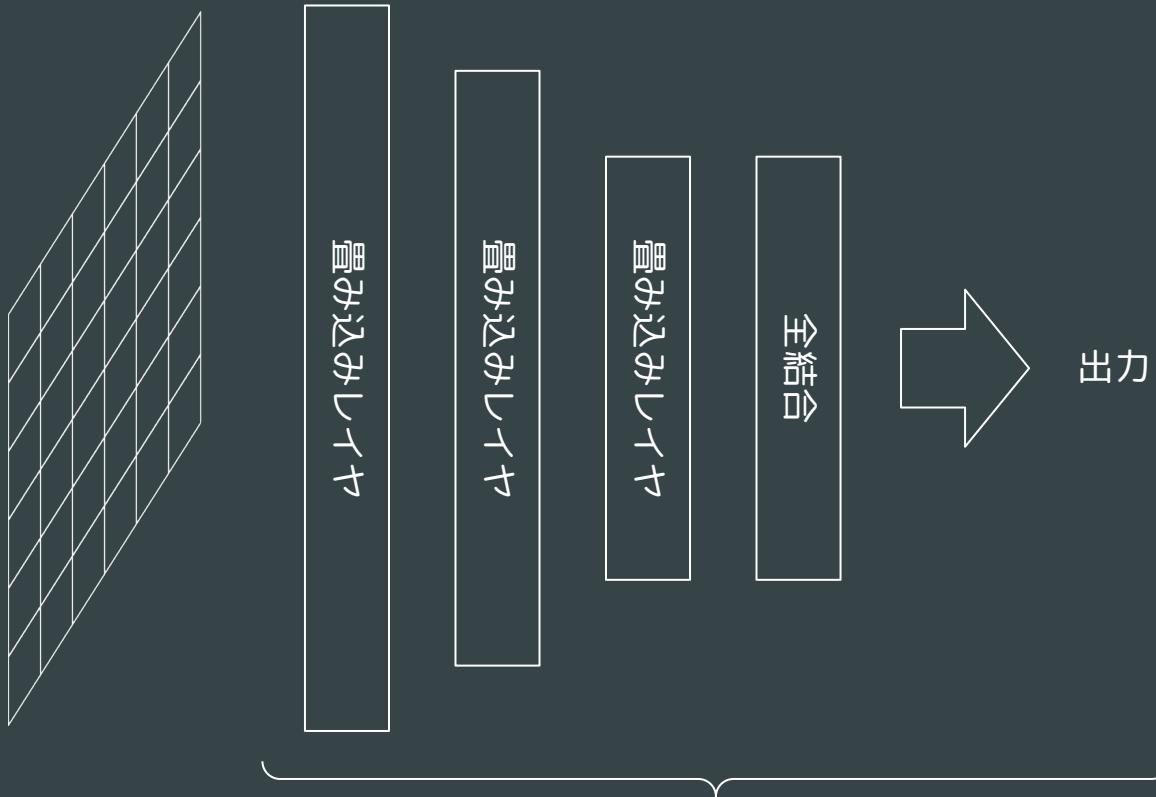
画像認識②



1つの畳み込みレイヤ

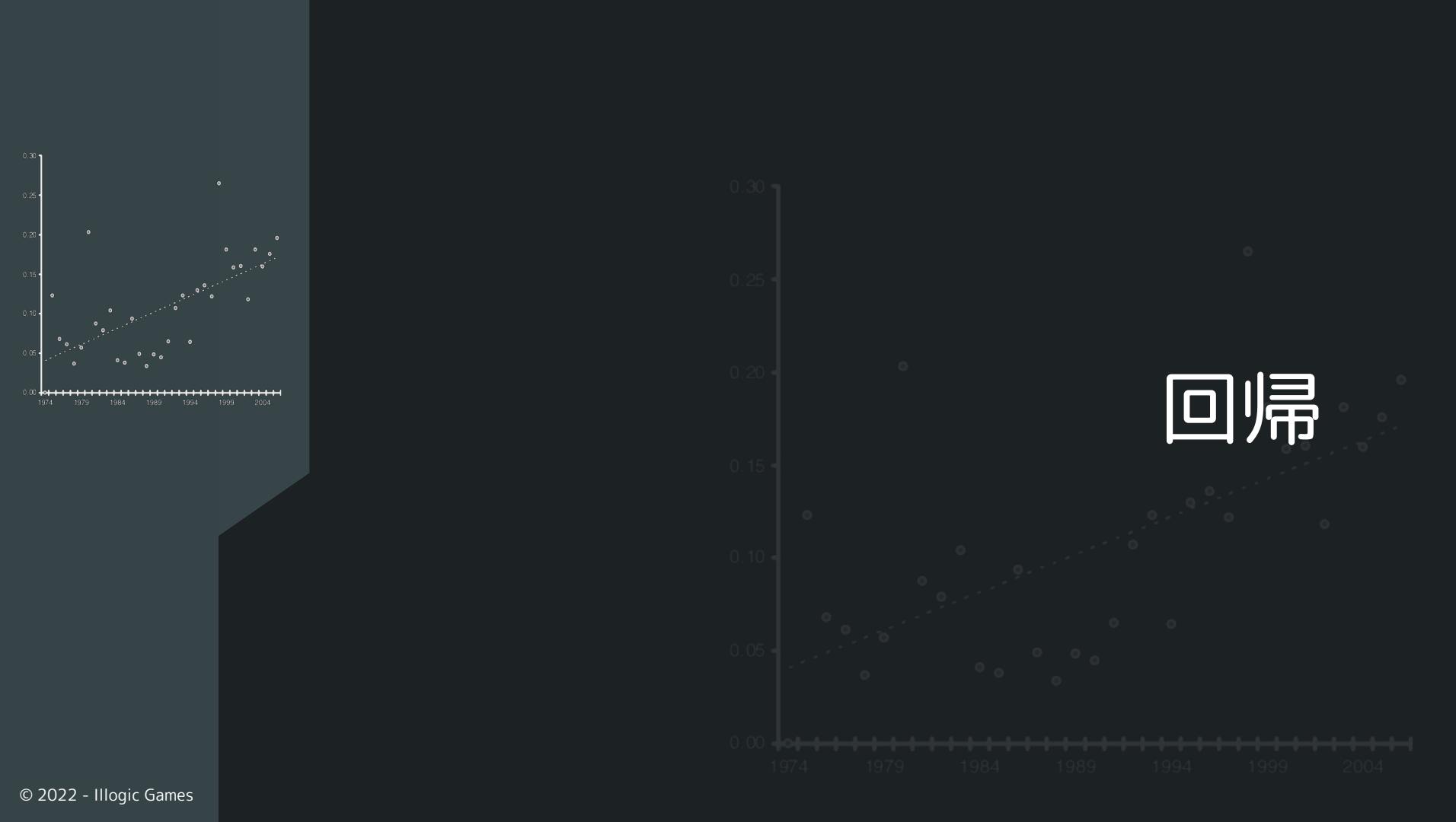


画像認識②



それでは、演習へ～





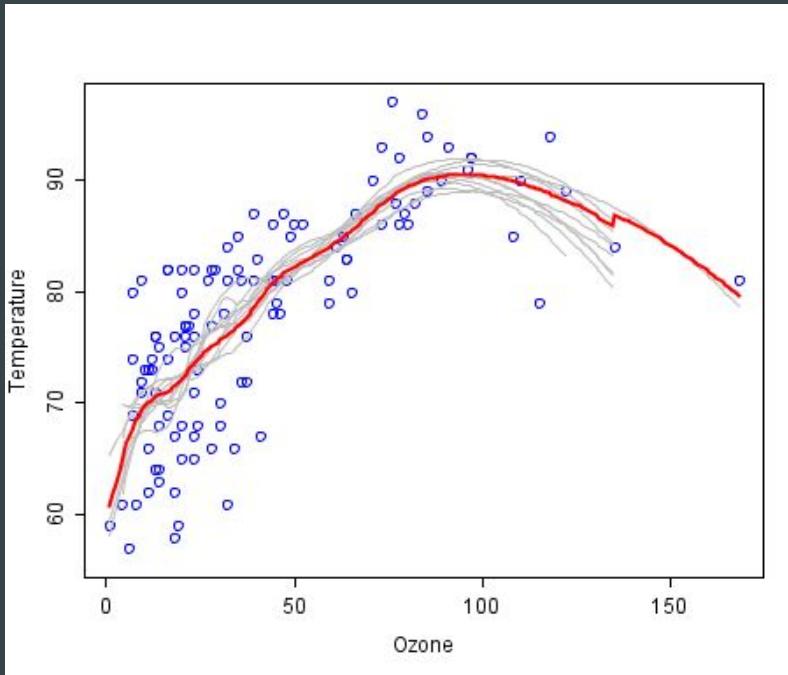
関数+入力 = 出力

$$f(x) = 2x + 5$$

$$x = 3$$

$$f(x) = 11$$

回帰：入力と出力データの関係性を求めたい



ニューラルネットワークは
入力と出力の関係性を求め、回帰できる！

分類や認識と異なり、望ましい
答えは離散ではない

MNISTの答え

0, 1, 2, 3, 4,
5, 6, 7, 8, 9

回帰課題の答え

0.251, 254.25,
3.1416, 1.3333, など



おまけ：ニューラル
ネットワークの落とし穴

オーバー・フィッティング (Overfitting)

- 学習の時に、loss（誤差）はほぼなく、精度は100%に近い
 - ただ、実際に使ってみると、全然正しくない
- 問題：学習データ覚えすぎ、汎用性がない
- 解決：
 - 学習データを増やす
 - 学習データを拡張する
 - Dropoutレイヤを使う
 - batch_sizeを大きくする

アンダー・フィッティング (Underfitting)

- どれぐらい学習さえても、loss（誤差）が減らなく、精度が上がらない
- 問題：ネットワークはデータから特徴を抽出できない
- 解決：
 - 前処理を考え直す
 - 中間層を増やす
 - loss関数を変える

バランス取っていないデータセット

- 学習の時、誤差が少なく、検証データでも問題ない
 - ただ、実際に使ってみると、ある特定のデータだけはうまく行かない
- 問題：そのデータは比較的に少ない
- 解決：各種のデータは大体同じ数があることを確認（アバウトで良い）

認知バイアス (cognitive bias)

結局、AIは、人間が学習したものである。したがって、人間、および社会のバイアスも含まれている。

① 顔認識の問題 (Gender Shades)

<http://gendershades.org/>

ニューラルネットワークの落とし穴

<http://gendershades.org/>



認知バイアス (cognitive bias)

結局、AIは、人間が学習したものである。したがって、人間、および社会のバイアスも含まれている。

① 顔認識の問題 (Gender Shades)

<http://gendershades.org/>

② アメリカの差別問題

<https://sitn.hms.harvard.edu/flash/2020/racial-discrimination-in-face-recognition-technology/>



まとめ

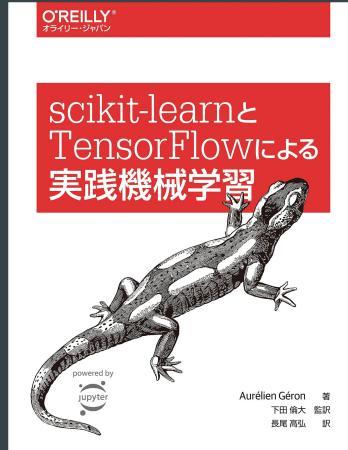
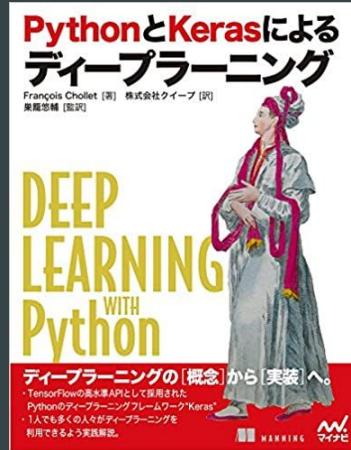
まとめ

- AIは万能の技術ではない
 - 特定の課題しか対応できない
 - データに強く依存する
- AIのエンジニアの仕事が大変！
 - データの前処理、後処理
 - ネットワークの設計、構築
 - Loss関数の選び方
 - パラメータの調整（学習率、unitsの数など）
- データ十分にあれば、面白い結果を得られる
 - 人間が気が付かないパターンも抽出できる！
- 様々な課題を解決できる
 - 分類、回帰は単純が、大幅に使うことができる
 - 疋み込みネットワークで、画像も処理できる

まとめ

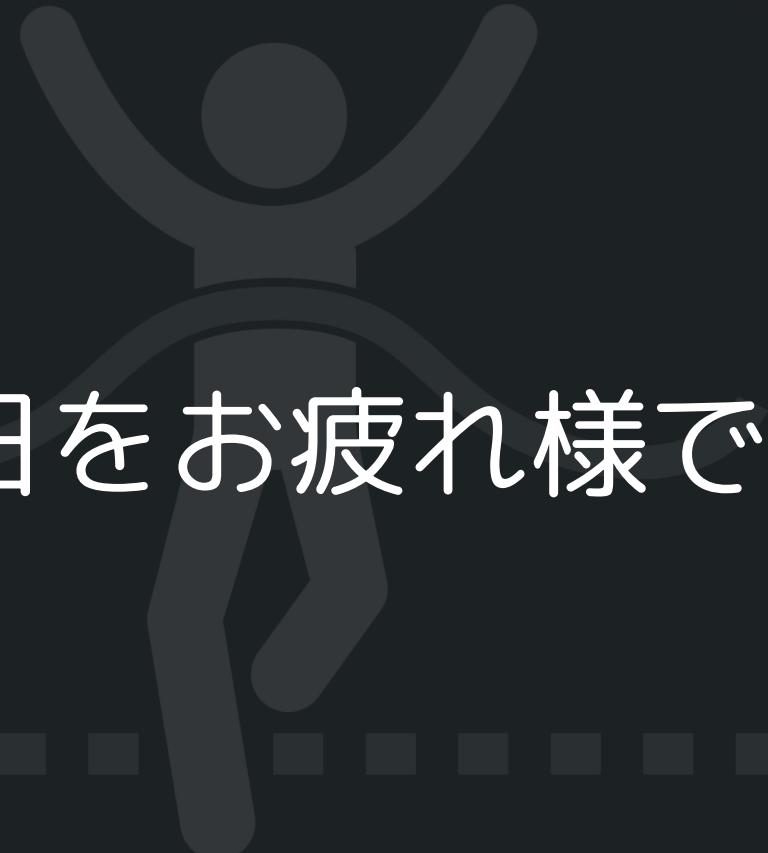
これから

- PythonとKerasによるディープラーニング
- scikit-learnとTensorFlowによる実践機械学習 (O'Reily)



- Kaggleを参加

- <https://www.kaggle.com/>



長い一日をお疲れ様でした